

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANITA PAES VINCENT

DETECÇÃO E AJUSTE DE CURVAS EM
IMAGENS DIGITAIS

RIO DE JANEIRO

2019

ANITA PAES VINCENT

DETECÇÃO E AJUSTE DE CURVAS EM
IMAGENS DIGITAIS

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Luziane F. de Mendonça, D.Sc.

RIO DE JANEIRO

2019

CIP - Catalogação na Publicação

V768d Vincent, Anita Paes
Detecção e ajuste de curvas em imagens digitais /
Anita Paes Vincent. -- Rio de Janeiro, 2019.
80 f.

Orientadora: Luziane Ferreira de Mendonça.
Trabalho de conclusão de curso (graduação) -
Universidade Federal do Rio de Janeiro, Instituto
de Matemática, Bacharel em Ciência da Computação,
2019.

1. Ajuste de curvas. 2. Processamento de imagem.
3. Computação gráfica. I. Mendonça, Luziane Ferreira
de, orient. II. Título.

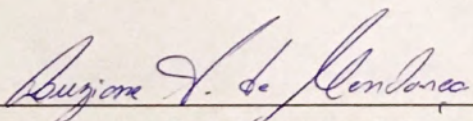
ANITA PAES VINCENT

DETECÇÃO E AJUSTE DE CURVAS EM
IMAGENS DIGITAIS

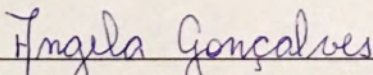
Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 14 de NOVEMBRO de 2018.

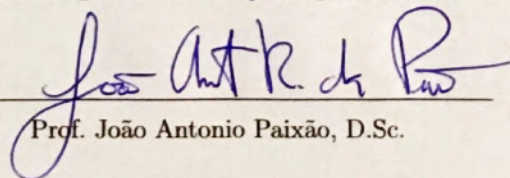
BANCA EXAMINADORA:



Prof. Luziane F. de Mendonça, D.Sc.



Prof. Angela Maria Silva Gonçalves, D.Sc.



Prof. João Antonio Paixão, D.Sc.

Para minhas avós e avô, com seu apoio e
carinho cheguei até aqui.

AGRADECIMENTOS

Agradeço à minha família, que me apoiou e incentivou a atingir cada conquista e que me permitiu crescer em um ambiente que valoriza o conhecimento além do amor. Sou grata aos meus pais, Roberto e Simone, por não pouparem esforços quando se tratou de me dar a melhor educação possível e à minha querida avó Luiza, que pagou parte dos meus estudos além de me incentivar e oferecer apoio por todo o caminho.

Obrigada a todos aqueles professores que têm paixão pelo ensino, vocês me inspiraram e ainda inspiram. O incrível trabalho de vocês faz diferença na vida de muitos alunos, inclusive a minha. Agradeço a todo o departamento do DCC, por me dar a oportunidade de me graduar em um curso de excelência apesar de todas as dificuldades que passa o ensino no Brasil.

Obrigada à Márcia Cerioli, minha orientadora acadêmica que se mostrou realmente presente e me ajudou quando eu precisava. E por último muito obrigada Luziane, por ter me orientado com todo o bom humor e paciência possível, pelas reuniões longas e pelas palavras de incentivo ou consolo e principalmente por ter me proporcionado a experiência de trabalhar em um projeto que eu amei.

RESUMO

A identificação e ajuste de curvas a partir de imagens digitalizadas é um problema relevante para diferentes propósitos como o estudo e pesquisa em ciências exatas e simulações. Este trabalho aborda o processo de extração de características de uma imagem para a identificação de uma curva e eixos, seguido do ajuste da curva identificada. Conseguindo como resultado final a fórmula matemática da função de ajuste assim como sua classe.

Palavras-chave: Processamento de Imagem. Esqueletização. Binarização. Ajuste de Curvas..

ABSTRACT

Curve fitting and identification based on a digitized image is a problem that is relevant to different purposes such as the study and research of mathematical sciences and simulation problems. This project regards the process of feature extraction on an image with the goal of identifying a curve and axis, followed by fitting the identified curve. Amounting to the mathematical formula of the function and the function class as the final result.

Keywords: Image Processing. Skeletonization. Binarization. Curve fit..

LISTA DE FIGURAS

Figura 1:	Imagem discretizada	17
Figura 2:	Vizinhança de um pixel [3]	17
Figura 3:	Conectividade de um componente	18
Figura 4:	Resultado da adição de três pontos diferentes (sobre a mesma reta) no acumulador [9]	20
Figura 5:	Visualização das transformações de reflexão, expansão e contração em um simplex [1]	23
Figura 6:	Caminho seguido pelo algoritmo de Powell [15]	24
Figura 7:	Processo de tratamento da imagem	25
Figura 8:	Comparação dos métodos de binarização em imagem sombreada . .	28
Figura 9:	Exemplo de Esqueletização [2]	28
Figura 10:	Esqueleto de imagem de alta resolução	29
Figura 11:	Esqueleto de imagem de baixa resolução	30
Figura 12:	Linhas de Hough marcadas (em azul) sobre a função (em branco) .	31
Figura 13:	Linhas agrupadas e classificadas	32
Figura 14:	Exemplo de três pares de reta verticais em que cada um expressa duas retas próximas. Os pontos se encontram em posições com pequena distância no eixo horizontal.	33
Figura 15:	Exemplo de interpolação	34
Figura 16:	Desenho original com eixos irregulares	35
Figura 17:	Remoção dos eixos e ruídos	36
Figura 18:	Imagem que não apresentou sucesso na detecção de dois eixos . . .	38
Figura 19:	Resultado do processamento de imagem com grade	39
Figura 20:	Resultados do ajuste polinomial	44
Figura 21:	Ajustes insatisfatórios de funções polinomiais	45
Figura 22:	Ajuste perfeito de funções exponenciais	47
Figura 23:	Ajustes insatisfatórios da classe exponencial	48
Figura 24:	Ajuste perfeito de funções logarítmicas	49
Figura 25:	Ajuste insatisfatório de funções logarítmicas	50
Figura 26:	Resultados de ajuste gaussiano com baixo valor de erro	51

Figura 27: Ajustes da Gaussiana com altos valores de erro	51
Figura 28: Ajuste perfeito da função racional	53
Figura 29: Ajuste insatisfatório da função racional	53
Figura 30: Ajuste perfeito da função seno	55
Figura 31: Ajustes insatisfatórios da função seno	55
Figura 32: Ajustes da função Mista	57
Figura 33: Ajuste com polinômio: $y = 1.6 * 10^{-10}x^5 - 3 * 10^{-7}x^4 + 0.0002x^3 -$ $0.015x^2 + 11.16x - 349.75$	66
Figura 34: Ajuste com polinômio: $y = 3.26 * 10^{-8}x^4 + 0.00004x^3 - 0.025x^2 +$ $6.9x - 406.75$	66
Figura 35: Ajuste com seno: $y = -97.8 * \sin(0.013x + 2.14) - 65.22$	67
Figura 36: Ajuste com polinômio: $y = 0.0038x^2 + 0.0063x - 0.644$	67
Figura 37: Ajuste com polinômio: $y = -4.32 * 10^{-8}x^4 - 3.7 * 10^{-6}x^3 -$ $0.0025x^2 - 0.06x + 82.84$	68
Figura 38: Ajuste com polinômio: $y = 0.0036x^2 - 0.36x + 63.42$	68
Figura 39: Ajuste com polinômio: $y = -0.00053x^2 + 1.01x - 21.52$	69
Figura 40: Ajuste com exponencial: $y = e^{-0.022*x+5.71} - 0.664$	69
Figura 41: Ajuste com polinômio: $y = 5.69 * 10^{-7}x^4 - 0.00014x^3 + 0.015x^2 -$ $0.076x + 40$	70
Figura 42: Ajuste com polinômio: $y = 1.17 * 10^{-7}x^4 - 0.00006x^3 + 0.015x^2 -$ $1.6x + 93.74$	70
Figura 43: Ajuste com seno: $y = -136.62 * \sin(-0.041x - 0.033) - 0.36$	71
Figura 44: Ajuste com seno: $y = -3.1 * \sin(-0.07x + 0.037) - 101.18$	71
Figura 45: Ajuste com seno: $y = 171 * \sin(0.016x + 0.0086) + 171$	72
Figura 46: Ajuste com seno: $y = -111.85 * \sin(0.027x - 0.86) + 111.25$	72
Figura 47: Ajuste com polinômio: $y = -3.07 * 10^{-16}x^8 + 5.4 * 10^{-13}x^7 - 3.97 *$ $10^{-10}x^6 + 1.57 * 10^{-7}x^5 - 0.00003x^4 + 0.005x^3 - 0.409x^2 + 18.1x - 98.13$	73
Figura 48: Ajuste com log: $y = -158 \log(-4.4x) + 832.6$	73
Figura 49: Ajuste com log: $y = 356.19 \log(2.69x) - 2167.28$	74
Figura 50: Ajuste com gaussiana: $y = \frac{1}{0.0024\sqrt{2\pi}} e^{\frac{(x-155.23)^2}{59.91^2}}$	74
Figura 51: Ajuste com racional: $y = \frac{1}{2.47*10^{-11}x^2 - 6.18*10^{-5}x + 1.33*10^{-5}}$	74

Figura 52: Ajuste com racional: $y = \frac{1}{1.65*10^{-6}x^2 - 5.14*10^{-4}x + 0.045}$	75
Figura 53: Ajuste com polinômio: $y = 2.3 * 10^{-11}x^5 - 9.7 * 10^{-9}x^4 - 3.4 * 10^{-6}x^3 + 0.0033x^2 - 0.98x + 102.29$	75
Figura 54: Ajuste com polinômio: $y = -3.07*10^{-16}x^8 + 5.4*10^{-13}x^7 - 3.97*10^{-10}x^6 + 1.57*10^{-7}x^5 - 0.00003x^4 + 0.005x^3 - 0.409x^2 + 18.1x - 98.13$	76
Figura 55: Ajuste com polinômio: $y = 2.99x + 134.67$	76
Figura 56: Ajuste com polinômio: $y = -0.021x^2 - 3.37x - 1.65$	77
Figura 57: Ajuste com polinômio: $y = 0.021x^2 - 3.02x - 185$	77
Figura 58: Ajuste com exponencial: $y = e^{0.0091*x+3.63} + 0.2$	78
Figura 59: Ajuste com exponencial: $y = e^{0.051*x+2.44} - 1.19$	78
Figura 60: Ajuste com polinômio: $y = -6.7*10^{-14}x^7 + 1.82*10^{-11}x^6 + 6.37*10^{-10}x^5 - 3.77*10^{-7}x^6 + 0.000011x^3 + 0.0008x^2 + 0.039x + 65.16$	79
Figura 61: Ajuste com polinômio: $y = 1.47*10^{-14}x^7 - 9.27*10^{-12}x^6 + 1.43*10^{-9}x^5 + 9.35*10^{-8}x^6 - 0.000028x^3 - 0.00004x^2 + 0.14x + 39.76$.	79
Figura 62: Ajuste com polinômio: $y = 1.65*10^{-14}x^7 - 1.66*10^{-11}x^6 + 5.26*10^{-9}x^5 - 2.79*10^{-7}x^6 - 0.00012x^3 + 0.02x^2 - 1.33x - 391.66$. . .	80
Figura 63: Ajuste com polinômio: $y = -5.26*10^{-18}x^8 + 4.53*10^{-15}x^7 - 8.84*10^{-14}x^6 - 7.45*10^{-10}x^5 + 1.09*10^{-7}x^6 + 0.00003x^3 - 0.005x^2 - 0.77x + 220.69$	80

LISTA DE TABELAS

Tabela 1: Ordem de complexidade estabelecida para cada classe	59
Tabela 2: Resultados Ajuste Global	61

SUMÁRIO

1	INTRODUÇÃO	14
2	CONCEITOS BASE	16
2.1	DISCRETIZAÇÃO DAS IMAGENS	16
2.2	CONECTIVIDADE	17
2.3	ALGORITMO FLOOD-FILL	18
2.4	DECLIVE (<i>SLOPE</i>)	18
2.5	TRANSFORMADA DE HOUGH	19
2.6	MÉTODOS NUMÉRICOS UTILIZADOS	21
2.6.1	Método dos Mínimos Quadrados	21
3	EXTRAÇÃO DE CARACTERÍSTICAS	25
3.1	PRÉ-PROCESSAMENTO	25
3.1.1	Filtro Bilateral	26
3.1.2	Limiarização por Mediana	26
3.1.3	Limiarização Adaptativa por Mediana	27
3.1.4	Comparação de Resultados	27
3.2	ESQUELETIZAÇÃO	28
3.2.1	Método de Esqueletização	29
3.2.2	Recorte da imagem	29
3.2.3	Avaliação e Reprocessamento	29
3.3	DETECÇÃO DOS EIXOS	30
3.3.1	Transformada de Hough	30
3.3.2	Classificação das Linhas	31
3.3.3	Agrupamento	32
3.3.4	Escolha dos Eixos	34
3.4	REMOÇÃO DOS EIXOS E RUÍDOS ESPÚRIOS	34
3.4.1	Seleção dos Pontos Iniciais	35
3.4.2	Testando a Conectividade	35
3.4.3	Metodologia para Remoção dos Eixos	36

3.4.4	Remoção de Pequenos Componentes Espúrios	36
3.5	RESULTADOS	38
4	CLASSIFICAÇÃO DA CURVA	40
4.1	CÁLCULO DOS PONTOS EM RELAÇÃO AOS EIXOS	40
4.2	AJUSTE POLINOMIAL	41
4.2.1	Método do Ajuste	42
4.2.2	Condicionamento	43
4.2.3	Evitando <i>Overfitting</i>	43
4.2.4	Resultados	44
4.3	AJUSTE EXPONENCIAL	45
4.3.1	Método do Ajuste	46
4.3.2	Resultados	46
4.4	AJUSTE LOGARITMO	47
4.4.1	Método do Ajuste	48
4.4.2	Lidando com Valores de Logaritmo Inválidos	48
4.4.3	Resultados	49
4.5	AJUSTE GAUSSIANO	50
4.5.1	Resultados	50
4.6	AJUSTE FUNÇÃO RACIONAL	51
4.6.1	Método do Ajuste	52
4.6.2	Resultados	52
4.7	AJUSTE SENOIDE	52
4.7.1	Método do Ajuste	54
4.7.2	Escolha de Valores Iniciais para o Algoritmo	54
4.7.3	Resultados	54
4.8	AJUSTE FUNÇÃO MISTA	55
4.8.1	Método do Ajuste	56
4.8.2	Discussão dos Resultados	56
4.9	AJUSTE GLOBAL	56
4.9.1	Método para a Escolha	57
4.9.2	Método de Prevenção de <i>Overfitting</i>	58

4.9.3	Discussão dos Resultados	59
5	CONCLUSÃO E TRABALHOS FUTUROS	62
	REFERÊNCIAS	64

1 INTRODUÇÃO

Funções matemáticas são ferramentas poderosas de modelagem e análise de fenômenos. Tanto a habilidade de desenho quanto a de interpretação de gráficos é parte do currículo obrigatório do estudante brasileiro e em diversas áreas se encontra a necessidade da identificação de funções, seja em análises estatísticas em laboratórios, em simulações físicas ou na matemática puramente teórica.

O objetivo deste projeto é, a partir da imagem digitalizada de uma curva, seja ela uma fotografia de um gráfico esboçado ou uma imagem gerada por computador, identificar a função matemática que melhor a representa.

Existem algumas ferramentas *open source* atualmente que permitem a extração semi-automática de dados a partir de gráficos. Dentre elas podem ser mencionadas o GetData [7], um digitalizador de gráficos que recebe como entrada um arquivo de imagem, além da escala (que deve ser entrada manualmente pelo usuário), e oferece como saída o conjunto de coordenadas dos pontos da curva. O Engauge Digitizer [5] é semelhante, com a vantagem de ser multi-plataforma e oferecer um ajuste simples para funções polinomiais, recebendo como entrada a imagem do gráfico e a localização dos eixos. Nenhuma plataforma faz o trabalho de extrair curvas de fotos ou imagens e fazer o ajuste comparativo para famílias variadas.

O projeto aqui implementado é uma grande ferramenta para estudantes buscando descobrir a fórmula de um gráfico, mas principalmente os desafios abordados neste trabalho têm grande potencial de expansão para aplicações de maior impacto; toda a parte de processamento de imagem e limpeza das curvas pode ser utilizada como base para limpeza e identificação de desenhos, números ou palavras. Enquanto a possibilidade da identificação de curvas e funções em vídeos e imagens de ambientes abre espaço para simulação e análise de problemas reais.

O texto é dividido em quatro capítulos além da introdução: o segundo capítulo introduz os conceitos chave para compreensão das atividades feitas ao longo do projeto e embasa o resto do texto; o terceiro capítulo descreve e ilustra todo o processo de processamento de imagem para obtenção dos dados relevantes a partir

das imagens de entrada. O capítulo quatro por sua vez apresenta as técnicas usadas para o ajuste da função a partir dos dados obtidos na etapa de pré-processamento de imagem. Ao final do capítulo quatro é feita uma discussão dos resultados.

Por último é feita a conclusão do trabalho e sugestões para trabalhos futuros. Todo o código descrito ao longo deste projeto foi implementado pela autora, fazendo uso de algumas rotinas disponíveis em Python para partes do processamento conforme ressaltado ao longo dos capítulos.

2 CONCEITOS BASE

Tanto o processamento de imagens quanto a classificação de curvas utilizam conceitos matemáticos e algoritmos específicos. Nesse capítulo são introduzidos e formalizados alguns desses conceitos, construindo uma base para os próximos capítulos.

2.1 DISCRETIZAÇÃO DAS IMAGENS

Conforme mencionado no Capítulo 1, a primeira parte deste trabalho versa sobre o desenvolvimento de uma sequência de etapas computacionais destinadas à manipulação de uma imagem digital. Aqui são introduzidos conceitos sobre como uma imagem é representada no formato digital e quais são seus atributos.

Uma imagem analógica é representada no meio digital através de uma matriz de *pixels*, ou seja, a imagem digitalizada é uma matriz bidimensional onde cada componente é a menor unidade representativa, ou seja, o pixel. [10]

Um pixel isolado expressa a cor daquela posição da imagem e tal informação pode ser representada através de números. Em imagens coloridas um único pixel guarda informações de intensidade de diferentes cores, nesse caso o pixel é um vetor. No formato *rgb*, por exemplo, um pixel guarda a intensidade das cores vermelho, verde e azul respectivamente, em uma escala que vai de 0 a 255. O computador usa tais valores discretos para renderizar a imagem na tela, atribuindo a cada pixel exibido a cor e intensidade certa, baseadas em seus dados. (Figura 1)

Com frequência optamos por converter imagens coloridas em *grayscale*, ou seja, em tons de cinza. Dessa maneira cada pixel guarda apenas uma informação: a intensidade daquele pixel na escala de 0 a 255, sendo 0 o preto absoluto e 255 o branco absoluto. Durante as operações feitas nos próximos capítulos também utilizamos imagens no formato binário, em que cada pixel guarda apenas 0 (preto) ou 1 (branco).

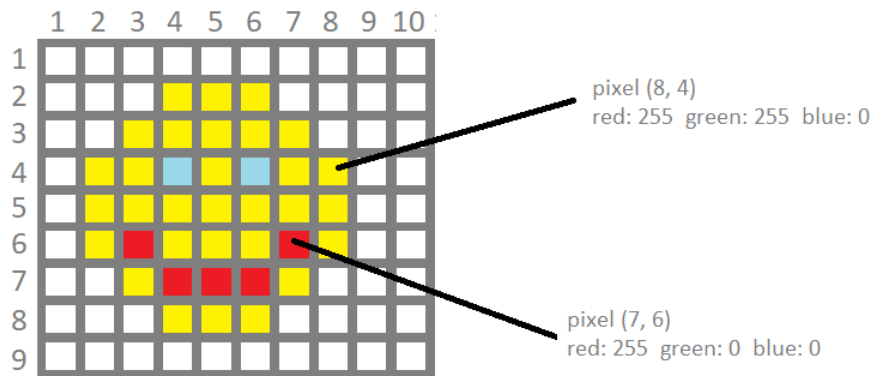


Figura 1: Imagem discretizada

2.2 CONECTIVIDADE

Durante a busca pela localização e extração dos pixels que representam os eixos cartesianos e a curva da função desenhada, os conceitos de vizinhança e conectividade são muito importantes. Tais conceitos são usados para a identificação de grupos de pixels como um único componente ou atributo.

O conceito de vizinhança estabelece quais pixels são considerados vizinhos a um determinado pixel. Analisando um único pixel podemos definir que ele está conectado a outro pixel em seus arredores se este for seu vizinho e tiver um valor parecido. Neste trabalho fazemos uso da vizinhança “8-connected”, o que quer dizer que os pixels adjacentes em todas as direções são considerados vizinhos. Se algum desses vizinhos for de intensidade parecida, quer dizer que estes dois pixels estão 8-conectados. [8]

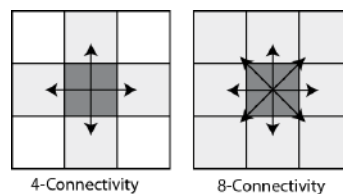


Figura 2: Vizinhança de um pixel [3]

De forma análoga, um componente na imagem pode ser considerado 8-conexo se, para cada um de seus pixels, é possível navegar entre vizinhos de intensidade parecida até se atingir todos os outros pixels, usando a regra de 8-conectividade

para a vizinhança. Conforme mencionado neste trabalho, sempre será considerada a conectividade de um componente como *8-conectividade*, como na figura 3.

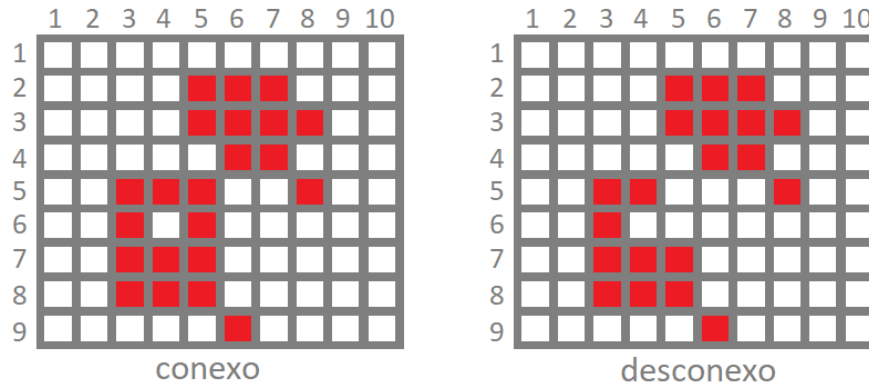


Figura 3: Conectividade de um componente

2.3 ALGORITMO FLOOD-FILL

Uma ferramenta de grande importância utilizada neste trabalho para identificar os pixels pertencentes aos eixos cartesianos e curva da função é o *flood-fill*.

O algoritmo de *flood-fill* tem como objetivo colorir todos os pixels de um elemento conexo, a partir de um pixel inicial de entrada, e da cor que se deseja colorir [17]. Existem diferentes implementações do algoritmo mas a ideia é sempre iniciar em um pixel e percorrer todos os pixels que são conexos a ele, os colorindo. Ele é muito útil para definir quais pixels fazem parte do mesmo componente conexo. O *flood-fill* recursivo pode ser definido pelo Algoritmo 1.

2.4 DECLIVE (*SLOPE*)

Um recurso utilizado na identificação de segmentos de retas como pertencentes aos eixos é a comparação do declive das retas analisadas.

O declive ou inclinação de uma reta expressa sua angulação em relação ao eixo das abcissas. O declive é calculado por:

Algorithm 1 Flood-fill

```

1: função FLOODFILL(pixel, cor_inicial, cor_nova)
2:   se pixel tem cor diferente a cor_inicial então
3:     retorna
4:   fim se
5:   pixel  $\leftarrow$  cor_nova
6:   para cada vizinho de pixel faça
7:     chama floodfill com pixel ▷ Chamada Recursiva
8:   fim para
9: fim função

```

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

onde (x_1, y_1) e (x_2, y_2) são dois pontos quaisquer distintos desta reta.

Um declive de 0 indica uma linha perfeitamente horizontal, declives com valores absolutos altos indicam uma reta próxima de vertical. Uma reta perfeitamente vertical possui declive indefinido pois o denominador se iguala a zero.

2.5 TRANSFORMADA DE HOUGH

Uma ferramenta que permitiu a localização das regiões conexas retas (como os eixos) foi a transformada de Hough.

A transformada de Hough é uma técnica de extração de características, utilizada em geral para a detecção de linhas. Nesse trabalho foi usada a Transformada de Hough probabilística, oferecida pela biblioteca OpenCV. [12]

A ideia principal da transformada de Hough consiste no fato de que uma linha pode ser representada por $\rho = x \cos \theta + y \sin \theta$ onde ρ denota a menor distância entre um ponto $P(x, y)$ da linha e a origem O , e θ é o ângulo formado entre o segmento OP e o eixo horizontal (complemento do ângulo de inclinação da linha). Sendo assim uma linha qualquer no espaço pode ser representada somente por (ρ, θ) .

Na transformada de Hough o primeiro passo é construir um acumulador, representando todos os possíveis pares (ρ, θ) de todas as possíveis linhas. Para cada ponto ativo na imagem (candidato a pertencer a uma linha) são simuladas todas as linhas que poderiam atravessá-lo em diversos ângulos. Para cada uma dessas possíveis linhas ρ e θ são marcados no acumulador. No final, os pares (ρ, θ) mais marcados no acumulador são as linhas localizadas, porque essas linhas são aquelas que cruzaram diversos pontos diferentes. (Figura 4)

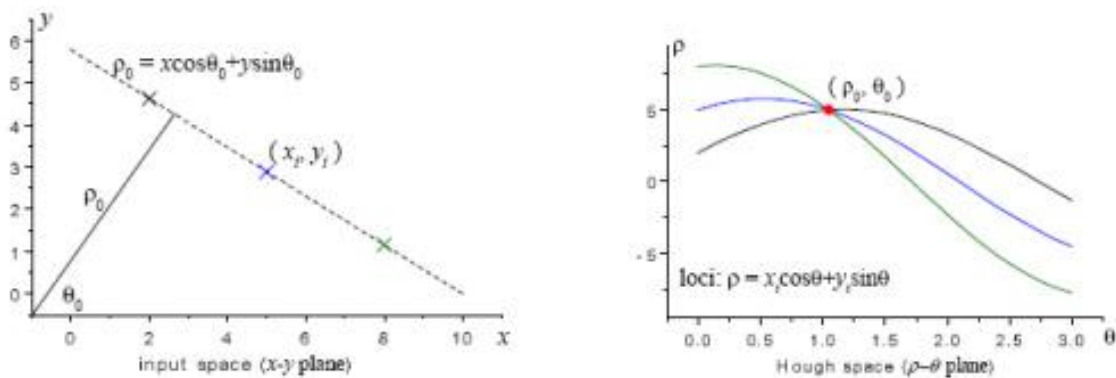


Figura 4: Resultado da adição de três pontos diferentes (sobre a mesma reta) no acumulador [9]

Isso fornece uma forma rápida de localizar todas as linhas na imagem. É preciso se estabelecer qual será a granularidade da simulação das linhas que passam em cada ponto. Como são testadas linhas com um certo ângulo de inclinação uniforme entre si, o tamanho desse ângulo é o que define a granularidade e nível de precisão da simulação. Também é preciso definir a quantos *pixels* de distância será testado um ponto; pode-se por exemplo utilizar todos os pixels da imagem como pontos, ou apenas testar pontos pulando um espaçamento uniforme entre eles.

A transformada de Hough *probabilística* é uma modificação da estrutura original onde, ao invés de serem avaliadas as linhas que cruzam todos os pontos, é determinado aleatoriamente um subconjunto dos pontos e a transformada de Hough é aplicada sobre essa amostra reduzida.

2.6 MÉTODOS NUMÉRICOS UTILIZADOS

Durante a etapa de classificação e ajuste de curvas foram utilizados diversos métodos numéricos para a aproximação da equação das curvas. Tais métodos serão discutidos nesta seção.

2.6.1 Método dos Mínimos Quadrados

O método dos mínimos quadrados é um método que pode ser usado para determinar a curva de melhor ajuste para um conjunto de pontos; o ajuste é feito realizando a minimização da soma dos quadrados obtida a partir de uma determinada função matemática. No caso do ajuste de funções, a ideia por trás do método dos mínimos quadrados é minimizar a distância entre uma função e os pontos que ela busca representar. [6]

O somatório dos mínimos quadrados pode ser representado por

$$S = \sum_{i=1}^n [y_i - f(x_i, \beta)]^2$$

onde (x_i, y_i) $i = 1 \dots n$ são os pontos para os quais a função deverá ser ajustada da melhor forma possível, f é a função de ajuste e β é o vetor de parâmetros utilizados na equação geral da classe de função escolhida.

O somatório pode ser minimizado de diferentes formas, e ao longo deste projeto três métodos foram utilizados, cada um se adequando melhor a uma ou mais classes de função. As próximas seções descrevem o funcionamento de tais métodos.

2.6.1.1 Minimização de Nelder-Mead

Nelder-Mead é um algoritmo para minimização baseado em *simplex* (politopo) e que utiliza busca heurística, sem em nenhum momento precisar calcular o gradiente da função (método *derivative-free*). O algoritmo começa definindo um simplex de $n + 1$ pontos x_j para a minimização de uma função com N variáveis, cada vértice é associado à função $f(x_j)$ sendo x_j o vetor representativo de cada ponto j no simplex.

Em seguida é feita uma sequência de transformações no simplex, com o objetivo de diminuir os valores das funções de seus vértices. A cada iteração o pior vértice será alterado; a transformação que será feita é determinada através do cálculo da função em um ou mais pontos de teste (localizados na direção do ponto médio dos demais vértices do simplex) seguida da comparação dos valores encontrados com os atuais valores dos vértices do simplex [11]. Os passos de processamento do algoritmo de Nelder-Mead são descritos a seguir.

1. **Ordenar os vértices:** determinar qual o melhor, segundo melhor e assim sucessivamente até o pior vértice, considerando o objetivo de minimizar a função $f(x_j)$.

2. **Calcular o centroide:** calcular o centroide como a média de todos os vértices exceto o pior, esse é o centroide oposto ao pior vértice.

3. **Transformação:** calcular o novo simplex. Primeiramente tentando substituir apenas o pior ponto através da reflexão, expansão ou contração do simplex na direção do centroide, gerando um vértice candidato que somente será aceito para substituir o pior vértice se sua imagem for menor do que a dele. Os pontos candidatos a novo vértice são todos testados na linha determinada entre o pior vértice e a centroide. Se isso falhar e o pior vértice não puder ser substituído por um ponto melhor então o simplex é encolhido na direção do melhor vértice.

O ciclo pode ser interrompido quando o simplex se torna suficientemente pequeno, no entanto o algoritmo nem sempre converge. Em geral implementações do Nelder-Mead incluem outros testes para avaliar o encerramento da execução.

A figura 5 ilustra as operações de reflexão, expansão e contração do simplex. O pior vértice v_3 é substituído. A substituição de v_3 por a ou por b representa uma contração. A substituição por d representa uma expansão e a substituição por c representa uma reflexão.

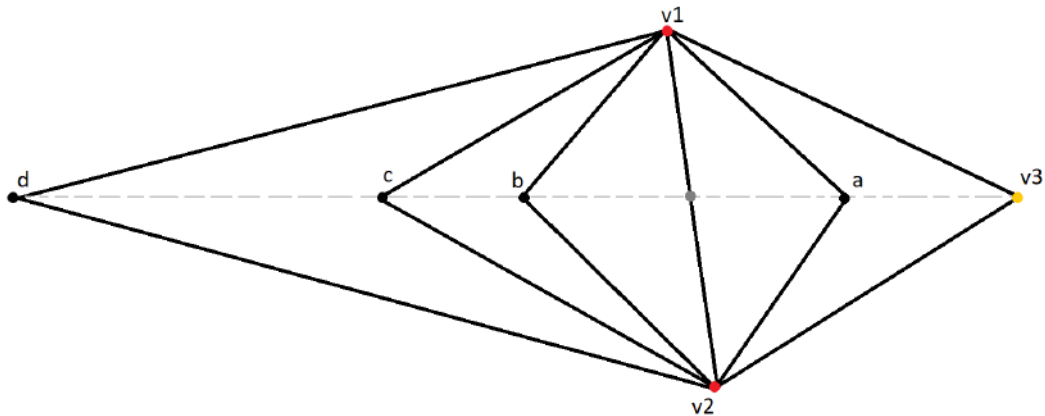


Figura 5: Visualização das transformações de reflexão, expansão e contração em um simplex [1]

2.6.1.2 Minimização de Powell

O método de minimização de Powell é um algoritmo que, assim como o método de Nelder-Mead, não utiliza gradiente para determinar a direção em que o mínimo deve ser procurado. O algoritmo envolve a definição de um conjunto de vetores que determina as direções nas quais o mínimo de $f(x, \beta)$ deve ser procurado, normalmente o conjunto vetorial utilizado é a base canônica, com dimensão igual ao vetor de parâmetros da minimização. [13]

Depois de definida a base e um ponto inicial que representa o primeiro valor testado para f , é feita a minimização unilateral; movimentos são feitos a partir do ponto inicial ao longo da primeira direção definida pela base até o seu ponto mínimo, a partir do qual a solução é investigada ao longo da segunda direção até se encontrar o mínimo nela, e assim por diante. Após a passagem por todo o conjunto vetorial, calcula-se a direção formada pelo ponto final deste processo e o ponto inicial. Esta nova direção, dita conjugada, substituirá a primeira da base. [11]

Esse ciclo de iterações é repetido indefinidamente até que não se possa fazer mais movimentos, o que significa que atingimos um mínimo. Uma única iteração

do algoritmo de Powell, com conjunto inicial de vetores \mathbf{u} que representa a base canônica na dimensão n , pode ser definida como:

1. Armazene a posição inicial P_0
2. Para $i = 1 \dots n$ mova P_{i-1} para o mínimo na posição u_i e chame essa posição de P_i
3. Para $i = 1 \dots n - 1$ determine $u_i \leftarrow u_{i+1}$
4. $u_n \leftarrow P_n - P_0$
5. Mova P_n para o mínimo na direção u_n e determine $P_0 \leftarrow P_n$

Esses passos são repetidos até que haja convergência. Uma visualização do caminho percorrido durante a execução do método de Powell pode ser visto na figura 6.

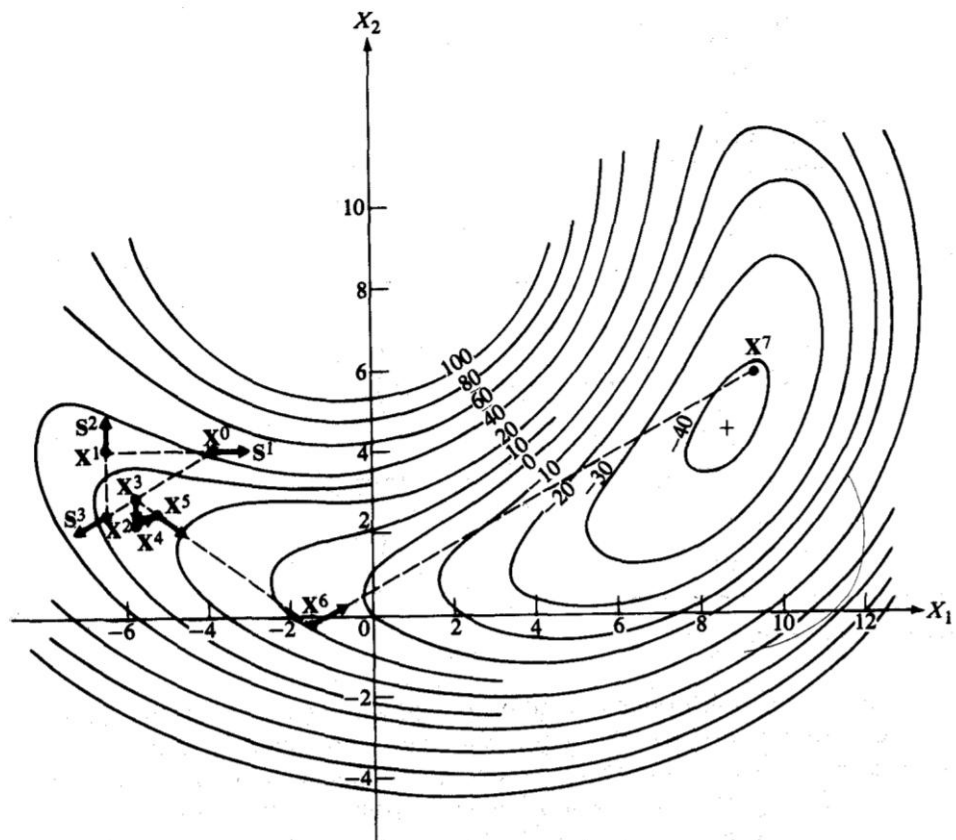


Figura 6: Caminho seguido pelo algoritmo de Powell [15]

3 EXTRAÇÃO DE CARACTERÍSTICAS

O primeiro passo para o reconhecimento de curvas a partir de imagens é o tratamento e processamento das imagens originais para a extração de características relevantes para o trabalho. O objetivo é diferenciar os pixels relevantes e ativos daqueles que representam o fundo da imagem (background), em seguida isolar os elementos da imagem que representam os eixos daqueles que representam a curva, e ao mesmo tempo remover pequenos ruídos que não façam parte do eixo ou da curva. A saída desejável desse processamento é que a curva seja representada por uma linha estreita, fornecendo uma entrada simplificada para a próxima etapa de processamento.

Toda a etapa de extração de características foi implementada em Python2.7. Todas as operações específicas de processamento de imagem como carregamento, manipulação de cores, conversão entre formatos e recorte de imagens foram feitas com a biblioteca *open source* OpenCV. [12]

A extração de características foi estruturada em cinco etapas - Pré-processamento, esqueletização, detecção dos eixos e remoção dos eixos e ruídos espúrios - as quais serão detalhadas individualmente nas próximas sessões.

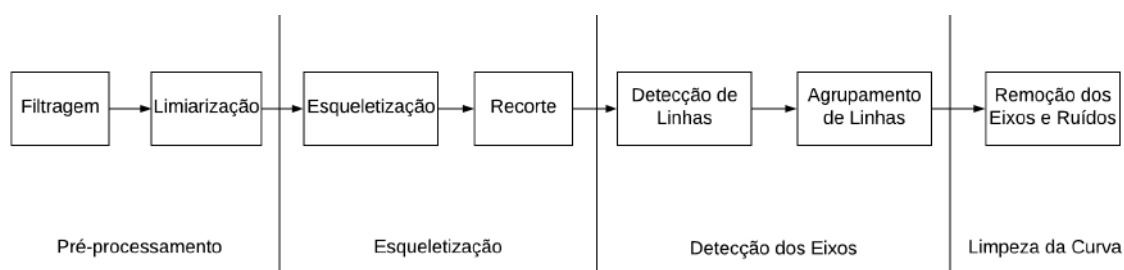


Figura 7: Processo de tratamento da imagem

3.1 PRÉ-PROCESSAMENTO

O pré-processamento envolve a remoção de ruídos aplicando filtros e operações de limiarização para a obtenção de uma imagem binária, que distingue entre áreas

de interesse e o fundo da imagem.

3.1.1 Filtro Bilateral

O filtro escolhido para remoção de ruídos foi o filtro bilateral. A escolha foi baseada na necessidade de preservar as bordas bem definidas das linhas e ao mesmo uniformizar o fundo, evitando que pequenas variações na intensidade do *background* fossem futuramente registradas como áreas de interesse no processo de binarização. [16]

O filtro bilateral é uma combinação de filtragem por domínio e por imagem. Essa combinação é definida por:

$$h(I)_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(|p - q|) G_{\sigma_r}(|I_p - I_q|) I_q$$

onde p é o pixel da imagem I , S é o conjunto dos pixels na vizinhança onde o filtro é aplicado, $\frac{1}{W_p}$ é o fator de normalização e G é o filtro Gaussiano, definido por:

$$G_{\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

σ_s na fórmula do filtro bilateral representa a janela de espaço onde é aplicado o *blur* (embaçamento), e σ_r representa a amplitude mínima valores de uma borda.

O filtro bilateral simplesmente causa o *blur* em grandes áreas espacialmente próximas e com valores de intensidade próximos. Preservando as estreitas áreas de borda em que há a mudança de intensidade. É um filtro que leva em consideração a posição e intensidade de cada pixel.

3.1.2 Limiarização por Mediana

Antes de começar o processo de binarização da imagem, primeiro ela é convertida em tons de cinza, caso seja colorida. Esses diferentes tons de cinza serão transfor-

mados em 0, representando o fundo (cor branca), ou 1, nossa área de interesse (cor preta).

O processo de limiarização por mediana envolve o cálculo da mediana dos tons da imagem; em seguida esse valor é usado para definir o limiar que divide valores entre dois grupos (0 ou 1). A regra é definida para cada pixel x :

$$p(x) = \begin{cases} 1, & \text{if } |p(x) - M| > thr \\ 0, & \text{caso contrário.} \end{cases} \quad (3.1)$$

onde M é a mediana e thr é o valor de *threshold* ou limiar definido.

3.1.3 Limiarização Adaptativa por Mediana

O método selecionado para binarização da imagem foi a limiarização adaptativa por mediana. Nesse método é feita uma limiarização por mediana em *cada* bloco de tamanho $N \times N$ (onde N é um valor fixo previamente escolhido), ao invés de ser calculada uma mediana global para toda a imagem. Isso evita que áreas sombreadas sejam completamente apagadas simplesmente por terem valores mais baixos que a mediana global.

Com a limiarização adaptativa foi possível identificar as linhas desenhadas, mesmo quando essas se encontravam em uma área sombreada, o que é importante considerando que as imagens de entrada podem ser uma fotografia com uma sombra por cima.

Após a realização de alguns testes numéricos, os valores que forneceram melhores resultados para as imagens teste foram: um tamanho de bloco N para limiarização de 50 pixels, e valor de *threshold* para a limiarização de 65.

3.1.4 Comparação de Resultados

Depois de feitos testes com várias imagens de complexidade variada, o uso da limiarização adaptativa produziu resultados superiores daqueles obtidos com a limia-

rização com mediana simples, especialmente em imagens com sombras (veja exemplo na figura 8).

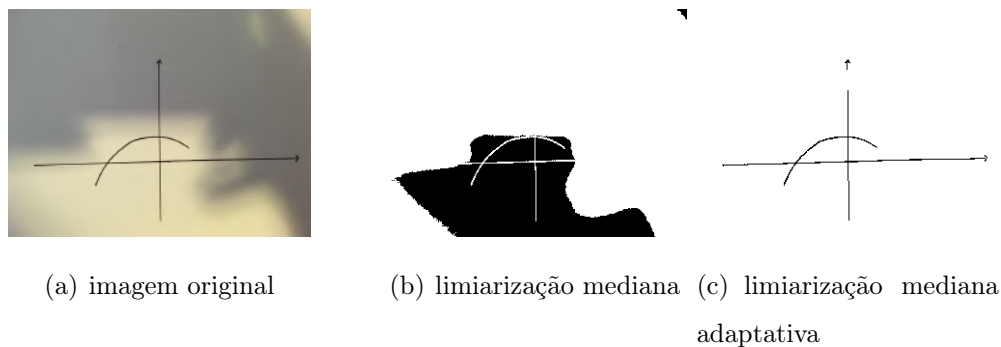


Figura 8: Comparação dos métodos de binarização em imagem sombreada

3.2 ESQUELETIZAÇÃO

Após obtermos a imagem tratada e binarizada, inicia-se o processo de esqueletização, reduzindo os elementos identificados da imagem a linhas estreitas. O esqueleto de um componente é a representação mínima dele, com apenas um pixel de espessura e o mais centralizado no componente possível. O esqueleto pode possuir laços e ramificações. (Figura 9)

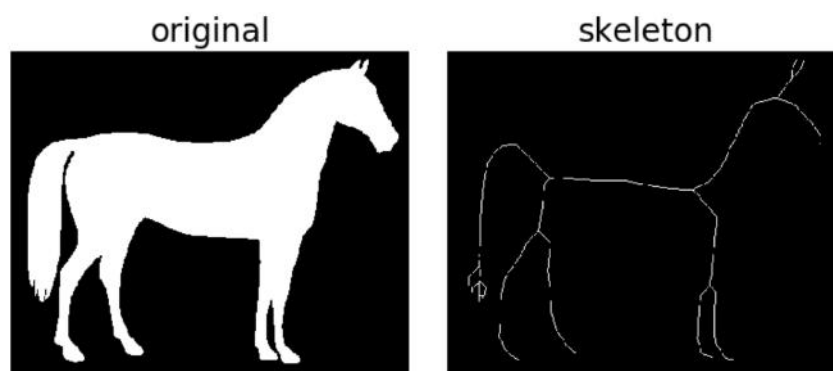


Figura 9: Exemplo de Esqueletização [2]

3.2.1 Método de Esqueletização

O esqueleto foi obtido com uma função pronta da biblioteca *Scikit Image*. [14] O método usado pela função é a passagem repetitiva pela imagem, retirando os pixels de borda nas áreas de interesse contanto que essa retirada não torne o componente não conexo. A repetição do processo é interrompida quando todos os componentes têm no máximo um pixel de espessura.

3.2.2 Recorte da imagem

As imagens de entrada podem ter um grande espaço composto apenas de valores iguais a zero nas laterais; na etapa de recorte nós retiramos tais áreas da imagem, obtendo somente o menor retângulo que contém nosso atual esqueleto. Essa imagem reduzida vai tornar os cálculos e avaliação dos resultados mais simples, uma vez que podemos trabalhar com menos dados. O resultado do processo de esqueletização e recorte pode ser visto nas figuras 10 e 11.

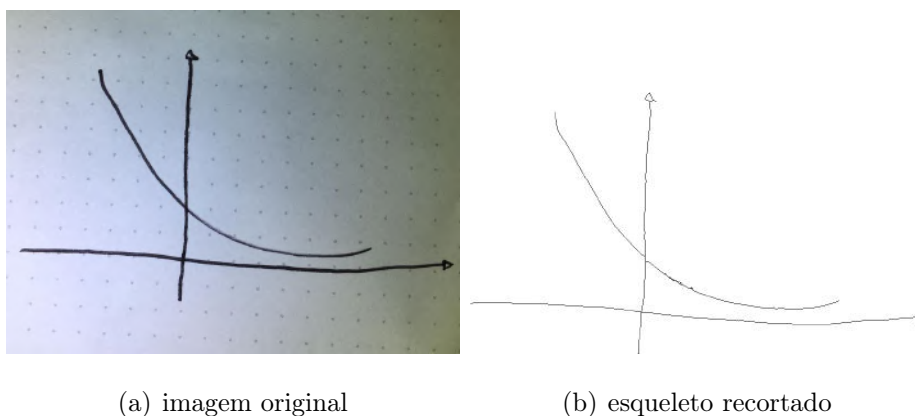


Figura 10: Esqueleto de imagem de alta resolução

3.2.3 Avaliação e Reprocessamento

Após a esqueletização e recorte da imagem, devemos avaliar o resultado do processo. Para isso avaliamos qual porcentagem da imagem é composta de pixels ativos, representando o esqueleto. Se esse valor for muito baixo (abaixo de 0.1%) isso quer

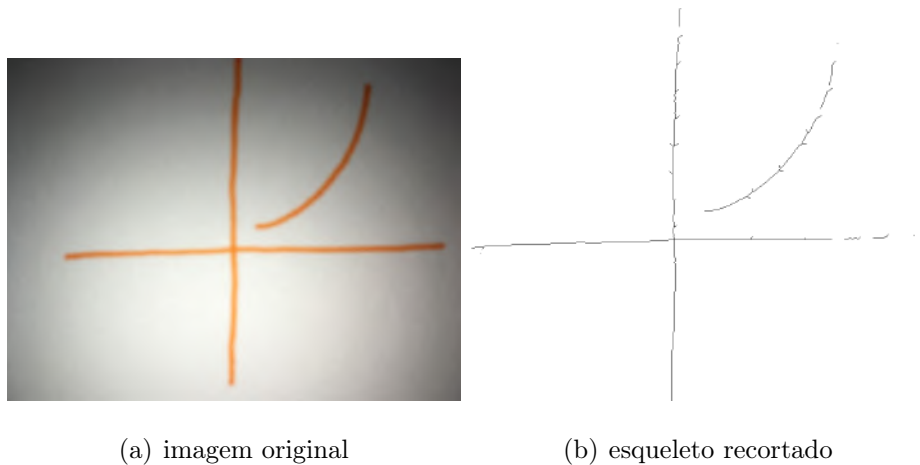


Figura 11: Esqueleto de imagem de baixa resolução

dizer que muita informação foi perdida na etapa anterior, a de pré-processamento. Nesse caso repetimos todo o processo, estabelecendo um *threshold* mais permissivo para a limiarização. Essa repetição acontece apenas uma vez, e é necessária em poucos casos, quando as linhas e o fundo da imagem têm intensidade parecidas.

3.3 DETECÇÃO DOS EIXOS

A partir da imagem simplificada pode-se começar a realmente retirar informações da mesma. Primeiramente foram localizados os trechos de linha na imagem, usando a transformada de Hough. Em seguida as linhas foram classificadas por seus ângulos e agrupadas, finalmente foram selecionadas quais linhas eram as melhores candidatas para representar os eixos X e Y .

3.3.1 Transformada de Hough

Usando a transformada de Hough (veja seção 2.5) foram encontrados todos os segmentos de reta na imagem. Considerando um tamanho mínimo para os segmentos e uma tolerância máxima de falhas nas linhas, o resultado foi um grande conjunto de pequenos segmentos de linha desconexos.

Esse processamento foi feito utilizando a Transformada de Hough Probabilística

do OpenCV, que permite a escolha de parâmetros para definir a precisão e tolerância de falhas da transformada.

A transformada de Hough probabilística tem como saída pequenos segmentos de reta que foram plotados sobre o esqueleto para visualização. (Figura 12)

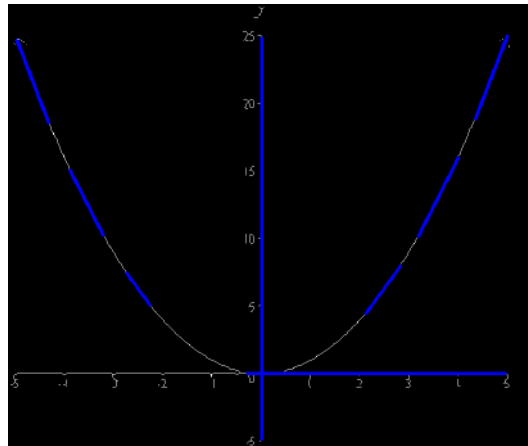


Figura 12: Linhas de Hough marcadas (em azul) sobre a função (em branco)

3.3.2 Classificação das Linhas

A partir dos segmentos de reta encontrados pela transformada de Hough, classificamos todas as linhas entre horizontais, diagonais ou verticais. Para isso foi usado o conceito de declive (seção 2.4).

Em nosso contexto as linhas classificadas como verticais são as linhas quase perfeitamente verticais (declive de valor alto) e as linhas classificadas como horizontais são as que são quase perfeitamente horizontais (declive próximo a zero). Todas as outras linhas são consideradas diagonais.

É indiferente o sinal do declive, porque não importa para que lado as retas estão pendendo, só nos importa a inclinação aproximada, horizontal ou vertical. Portanto para cada linha foi calculado o declive. Se o módulo do declive for igual a zero ou próximo de zero, a linha é considerada horizontal. Se o módulo do declive for acima de 7.5 a linha é considerada vertical. Nos casos em que a linha é perfeitamente vertical é impossível de se calcular o declive; nesse caso verificamos se a abscissa é constante nas duas extremidades da linha, indicando que é vertical.

Todas as linhas que não são horizontais nem verticais são consideradas diagonais; tais linhas são descartadas no processamento daqui para frente.

Na figura 13 as linhas foram coloridas de acordo com sua classificação, as linhas verdes são as horizontais, as vermelhas são verticais e as azuis são as linhas consideradas diagonais.

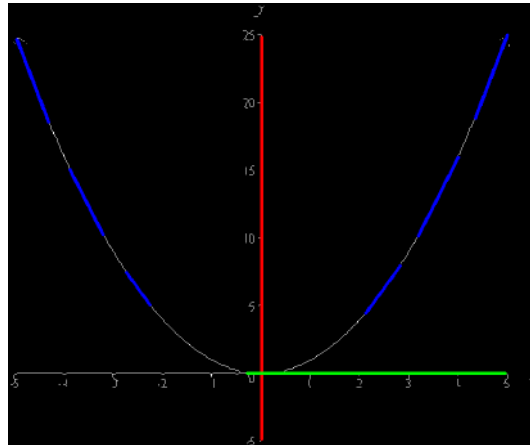


Figura 13: Linhas agrupadas e classificadas

3.3.3 Agrupamento

Após a classificação das linhas e o descarte das retas diagonais ainda sobram diversos segmentos de reta espalhados. Os segmentos que representam aproximadamente a mesma reta serão agrupados em uma reta única, resumindo as informações obtidas em um número menor de segmentos mais longos.

O agrupamento começa a partir de um dos segmentos de reta; a partir dessa reta todas as outras são avaliadas, e se estiverem próximas são interpoladas. Ao final, essas retas são substituídas pela reta única obtida pela interpolação do grupo.

3.3.3.1 Critério de Proximidade

O critério usado para definir se duas retas são “próximas” foi:

1. As duas tem a mesma direção, seja esta diagonal ou horizontal

Algorithm 2 Agrupamento de segmentos

```

1: função GROUP(conjunto)
2:   para cada reta em conjunto faça
3:      $retas\_proximas \leftarrow acha\_proximas()$ 
4:      $retas\_unica \leftarrow interpola(reta, retas\_proximas)$ 
5:     remove reta do conjunto
6:     remove retas_proximas do conjunto
7:     adiciona reta_unica no conjunto
8:   fim para
9: fim função

```

2. No caso em que as duas retas são verticais, nós analisamos seus pontos extremos. Se espera que os pontos variem muito quanto a posição no eixo Y por as retas serem verticais. Mas se os pontos extremos estiverem em uma posição parecida no eixo X, as retas são consideradas próximas.
3. Se as duas forem horizontais elas devem ter todos os pontos extremos em posições próximas no eixo Y.

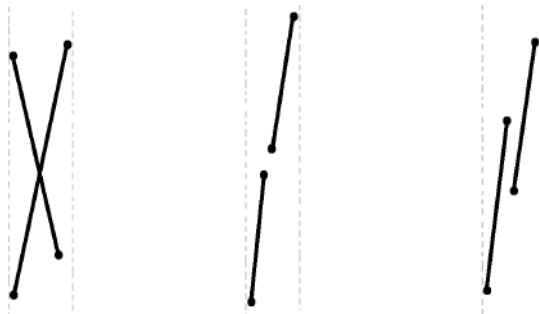


Figura 14: Exemplo de três pares de reta verticais em que cada um expressa duas retas próximas. Os pontos se encontram em posições com pequena distância no eixo horizontal.

3.3.3.2 Método de Interpolação

O método de interpolação escolhido foi buscar os pontos mais extremos e opostos das duas linhas e conectar eles, formando uma nova linha (Figura 15). Para

interpolar diversas linhas o processo é feito por partes.

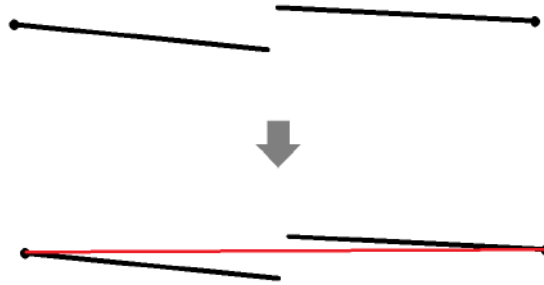


Figura 15: Exemplo de interpolação

3.3.4 Escolha dos Eixos

Depois do agrupamento se espera que só sobrem duas retas, representando o eixo X e Y. Mas isso nem sempre é verdade, outras linhas horizontais e verticais na imagem podem ficar após o processamento. Portanto estabelecemos um processo de desempate, e a partir dele escolhemos quais linhas serão consideradas os eixos. Para cada par de linhas com orientação diferente, é calculado o ângulo entre elas. O par de linhas mais ortogonal, ou seja, com o ângulo mais próximo de 90° , é escolhido.

Ao final desse processo temos duas linhas que representam o eixo e a partir de suas fórmulas é calculado também o ponto da origem do espaço.

3.4 REMOÇÃO DOS EIXOS E RUÍDOS ESPÚRIOS

Depois de localizadas as retas que representam os eixos, a próxima etapa é remover as linhas na imagem que façam parte desses eixos. Essa tarefa não é simples pois o desenho dos eixos nem sempre segue uma reta perfeita; as retas localizadas na etapa anterior não se encontram exatamente sobre todo o desenho original do eixo X e Y, como ilustrada na figura 16

Foi formulada uma estratégia para esse objetivo específico: usando as retas obtidas na etapa anterior como guia removeremos o desenho dos eixos, evitando ao

máximo retirar áreas do desenho que possam representar a curva. Em seguida, por um processo simples, serão removidos quaisquer pequenos pedaços de reta que possam ter sobrado; tais pequenos fragmentos em geral são apenas ruído oriundo de informações que não fazem parte da curva nem dos eixos. Depois dessa série de remoções, tudo aquilo que sobrar na imagem será considerado parte da curva, nosso objeto de estudo.

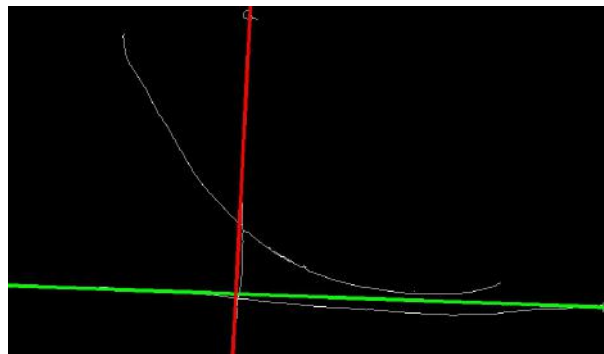


Figura 16: Desenho original com eixos irregulares

3.4.1 Seleção dos Pontos Iniciais

A remoção de cada eixo precisa se iniciar em um ponto que esteja sobre o eixo. Para garantir isso os pontos de início do percurso são os pontos extremos das retas que achamos anteriormente para definir os eixos.

3.4.2 Testando a Conectividade

Para percorrermos o traçado dos eixos é preciso definir quais são os pontos próximo de um ponto *pivot* que estão conectados a ele. Para identificar isso facilmente, primeiro é definido um “quadrado” em volta do ponto englobando toda a área que seria próxima dele. Depois é usado o algoritmo *floodfill* (veja seção 2.3), a partir desse ponto que tem como limite o quadrado, não podendo continuar a colorir além dele. O resultado é que os pontos coloridos são os pontos próximos e conexos ao ponto *pivot*. Uma implementação do floodfill é fornecida pelo OpenCV.

3.4.3 Metodologia para Remoção dos Eixos

Para a remoção dos eixos a estratégia foi começar de um ponto e ir se movendo para cima (no caso do eixo vertical) enquanto tiver algum ponto conexo em um ângulo razoável e ao mesmo tempo ir apagando o caminho percorrido. O mesmo processo depois foi repetido, mas se movendo para baixo.

Isso foi repetido nos dois pontos iniciais de cada eixo, no caso do eixo horizontal a movimentação foi para a direita e esquerda. Esse método permitiu apagar os eixos mesmo quando esses eram muito imperfeitos, sem seguir exatamente a reta achada anteriormente; ao mesmo tempo as linhas que potencialmente pertenciam à curva foram poupadas de muita interferência.

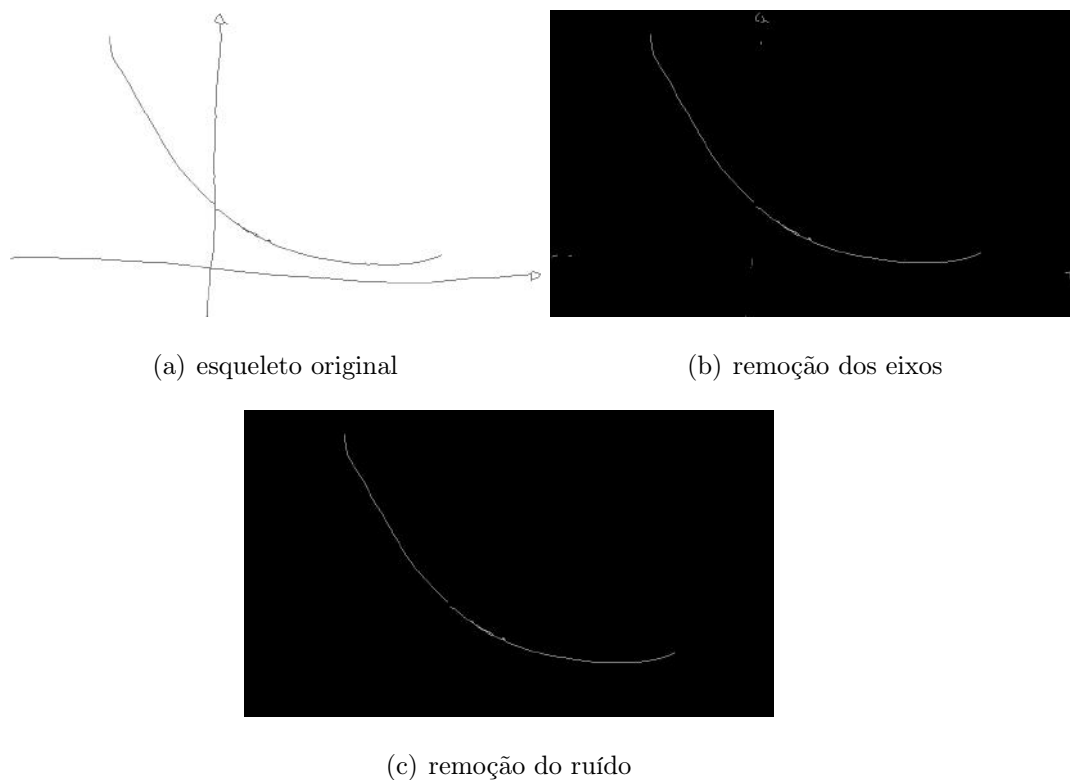


Figura 17: Remoção dos eixos e ruídos

3.4.4 Remoção de Pequenos Componentes Espúrios

Depois da remoção dos eixos o que sobra são a curva e pequenos elementos que possam ter permanecido. É importante notar que a curva pode estar desconexa,

Algorithm 3 Operação de remoção a partir de um dos pontos do eixo Y

```

1: função REMOVE(ponto_inicial, reta_do_eixo)
2:   respeitando a fórmula da reta se move alguns pixels para cima
3:   se não estiver em um pixel branco busque um nas proximidades
4:   ponto  $\leftarrow$  ponto_inicial
5:   enquanto verdadeiro faça
6:     pontos_conexos  $\leftarrow$  coloridos_floodfill_limitado(ponto)
7:     proximo_ponto  $\leftarrow$  ponto_conexo_mais_acima
8:     se angulo(ponto, proximo_ponto) é aceitavel então
9:       ponto  $\leftarrow$  proximo_ponto
10:    senão
11:      encerra while
12:    fim se
13:  fim enquanto
14:  repete o mesmo se movendo para baixo
15: fim função

```

dividida em segmentos. Os pequenos componentes serão considerados ruído e podem ser oriundos de desenhos de seta no final dos eixos, desenhos de números sobre os eixos, marcações no fundo ou outras coisas escritas.

Para remover os elementos pequenos são localizados todos os diferentes componentes conexos da imagem assim como suas áreas. O componente de maior área (excluindo o fundo) é o melhor candidato para ser parte da curva. Todos os outros elementos que tenham menos de 20% do tamanho do maior elemento são eliminados.

Com isso conseguimos uma curva limpa, sem ruídos e sem os eixos (figura 17). Estes pontos que compõe a curva, as retas que representam os eixos e o ponto de origem (a interseção entre as duas retas) serão a entrada para a próxima fase, a fase de aproximação das curvas.

3.5 RESULTADOS

As 32 imagens de teste foram divididas em três grupos, o primeiro grupo é de imagens “simples”, sem linhas ou pontos no fundo. O segundo grupo é de imagens com quadriculado e pontilhado no fundo, mas com a curva e eixos representados com uma cor mais forte. O terceiro grupo é composto de gráficos com quadriculado no fundo, em que o quadriculado tem intensidade igual ou parecida à curva e os eixos.

O grupo 1 e 2 responderam muito bem ao pré-processamento, de 27 imagens apenas uma não pôde ser tratada. Ela pode ser vista na figura 18, o algoritmo não foi capaz de identificar dois eixos neste caso, apenas um.

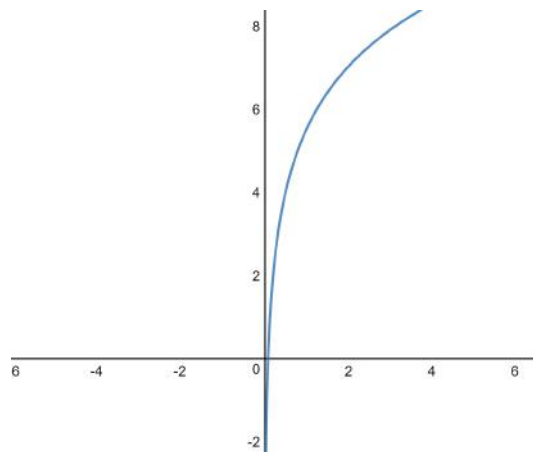
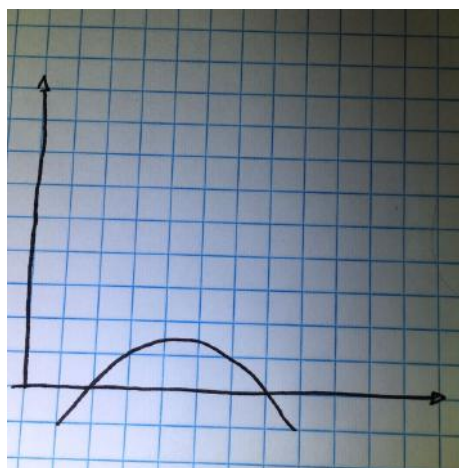


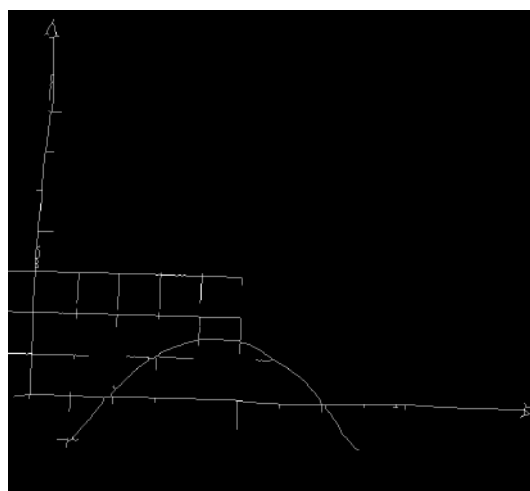
Figura 18: Imagem que não apresentou sucesso na detecção de dois eixos

No grupo 3, composto por 5 imagens, apenas uma respondeu bem ao pré-processamento, se mostrando limpa e sem linhas após o processo. A figura 19 mostra uma das figuras do grupo 3 e o resultado de seu processamento.

As imagens que não responderam ao pré-processamento produziram resultados ruins na fase de ajuste, ou não puderam ser ajustadas de forma alguma, no caso de eixos não encontrados.



(a) imagem original



(b) resultado da extração de características

Figura 19: Resultado do processamento de imagem com grade

4 CLASSIFICAÇÃO DA CURVA

Depois de feito o processamento da imagem original descrito no capítulo 3 temos as informações de base para começar a classificação das curvas. Os dados que precisamos são as posições dos pontos da curva na imagem e a posição da origem do sistema cartesiano (interseção dos eixos).

A primeira coisa que precisa ser feita é definir o valor dos pontos da curva **em relação à origem**; o único valor que temos no momento é a posição de cada pixel marcado na imagem, em relação ao canto esquerdo superior. Para fazer o ajuste da curva esses valores dos pontos em relação ao eixo cartesiano precisam ser definidos, conforme será descrito na seção 4.1.

Tomando por base esses pontos é realizada uma busca pela função que melhor representa esses dados dentro de uma classe definida, foram utilizadas as seguintes classes de função: polinomial (de grau 0 a 8), exponencial, logarítmica, senoide, racional e gaussiana; conforme descrito nas seções 4.2 a 4.7. Apesar de não ser um grupo muito vasto, a opção por essas famílias de funções deve-se ao fato de serem bastante conhecidas e utilizadas. Além destas foi feito o ajuste da função formada pela soma de todas as classes mencionadas, denominada neste projeto de função *mista*; o ajuste da função mista teve resultados pouco satisfatórios que serão discutidos na seção 4.8.

Por fim, para cada uma das imagens de teste, foi feito um ajuste global. Comparando os ajustes de todas as diferentes classes e dando apenas um resultado com a classe de função e a equação de melhor ajuste para cada imagem.

Cada etapa do processo é descrita ao longo desse capítulo, assim como é apresentada também uma discussão dos resultados e desafios.

4.1 CÁLCULO DOS PONTOS EM RELAÇÃO AOS EIXOS

Para conseguir as coordenadas dos pontos em relação ao eixo cartesiano precisamos fazer uma operação sobre cada um dos pontos. Em imagens a posição dos

pixels é medida de cima para baixo no eixo Y e da esquerda para direita no eixo X. Isso significa que o sinal no eixo Y deverá ser invertido ao fazer a conversão.

Portanto a operação feita para cada ponto é

$$x_i \leftarrow x_i - O_x$$

$$y_i \leftarrow -(y_i - O_y)$$

onde (O_x, O_y) são as coordenadas da origem determinadas durante o pré-processamento e x_i, y_i são as coordenadas do i-ésimo ponto tabelado. Seria relevante nesse momento dividir a posição dos pontos pela escala do gráfico; no entanto a determinação da escala de um gráfico apenas a partir de seu desenho é um trabalho extenso, não cabendo no escopo desse projeto. Por esse motivo a medida da posição dos pontos é feita em **pixels**, ignorando possíveis escalas.

Após esse processamento inicial de todos os pontos, usaremos os valores calculados em todas as operações de ajuste subsequentes.

4.2 AJUSTE POLINOMIAL

O ajuste da função polinomial foi feito para polinômios de graus de 0 a 8. A expressão geral da polinomial escolhida foi:

$$y = \sum_{k=1}^n a_k x^k$$

onde n é o grau do polinômio e $(a_1 \dots a_n)$ é o vetor de parâmetros. Para fazer o ajuste da função nosso objetivo é minimizar o quadrado das diferenças entre a imagem de cada ponto tabelado de acordo com o polinômio e as imagens dos pontos de entrada, como descrito no capítulo 2; a função que buscamos minimizar é portanto

$$q(a) = \sum_{i=1}^N \left[y_i - \sum_{k=1}^n a_k x_i^k \right]^2$$

onde os (x_i, y_i) são cada um dos pares que representam os pontos de entrada. O problema da minimização do quadrado das diferenças para ajuste polinomial é tipicamente resolvido solucionando a equação

$$\nabla q(a_1, a_2, \dots, a_k) = 0$$

que resulta no problema

$$V(x) * a = y$$

sendo $V(x)$ a matriz de Vandermonde gerada por x_1, \dots, x_n e a o vetor dos coeficientes. A partir daí a determinação dos coeficientes minimizadores se reduz à solução do sistema linear.

4.2.1 Método do Ajuste

O ajuste foi feito com o auxílio da função *polyfit* disponibilizada na biblioteca numpy do python2.7. Essa função calcula os mínimos quadrados e minimiza o erro (o quadrado das diferenças), retornando os coeficientes minimizadores, além do erro final encontrado.

O *polyfit* faz o ajuste polinomial a partir da minimização do somatório dos mínimos quadrados **com peso**. Buscando minimizar o erro

$$E(a) = \sum_{i=1}^N w_i^2 * \left[y_i - \sum_{k=1}^n a_k x_i^k \right]^2$$

sendo cada peso w_i determinado internamente pelo próprio *polyfit*. A minimização é feita pela resolução do sistema linear, como descrita no item anterior, fazendo uso da decomposição *SVD* para a resolução.

Foi feito o ajuste para cada um dos graus em sequência, começando pelo grau 0 e subindo até encontrar um ajuste adequado. O método de escolha do melhor ajuste é discutido na seção 4.2.3.

4.2.2 Condicionamento

O principal desafio no ajuste da função polinomial foi o mal condicionamento da matriz do problema linear, causado pela extrema diferença na magnitude dos valores na matriz a medida que as potências aumentam.

Por esse motivo a resolução simples do sistema linear, inicialmente implementada, não foi o suficiente para o ajuste, mostrando resultados negativos à medida que funções de maior grau eram testadas. A rotina *polyfit* foi capaz de fazer o ajuste, contornando o problema de mal condicionamento através da inclusão de pesos ao problema dos mínimos quadrados.

4.2.3 Evitando *Overfitting*

Ao compararmos diferentes graus de polinômios buscando aquele de melhor ajuste é comum que o polinômio de resposta seja de grau alto, mesmo quando um polinômio de grau menor é o suficiente para fazer um ótimo ajuste da função. A flexibilidade de polinômios com graus mais altos faz com que eles se ajustem exatamente aos pontos de entrada, mesmo quando esses pontos apresentam pequenas distorções ou erros, esse fenômeno é chamado de *overfitting*. Foram utilizados dois critérios diferentes para prevenir o overfitting.

Primeiramente o cálculo do erro de cada ajuste foi feito na ordem do menor para o maior grau. A qualquer momento se o erro ficasse abaixo de uma limiar que representa um ajuste quase perfeito a execução era interrompida e o atual grau escolhido como candidato; esse limiar foi estabelecido como quatro vezes o número de pontos de entrada.

Após a escolha do candidato inicial era feita uma comparação, avaliando se algum grau menor apresentou um valor de erro próximo do encontrado no melhor grau. Se algum grau mais baixo com erro parecido fosse encontrado, esse grau era o novo escolhido.

Essa abordagem apresentou bons resultados, reduzindo o grau do polinômio es-

colhido para as imagens de entrada, com pouca influência na qualidade dos ajustes.

4.2.4 Resultados

O ajuste da função polinomial apresentou excelentes resultados, conseguindo erro baixo em quase todas as curvas, apesar dos graus das equações escolhidas, em geral, terem sido mais altos que o esperado, especialmente em curvas esboçadas à mão.

A figura 20 mostra o gráfico da função encontrada pelo ajuste em dois casos em que o ajuste foi quase perfeito. O gráfico foi esboçado sobre o desenho inicial da curva para melhor visualização. Todos os gráficos de resultados apresentados neste trabalho mostram a equação e parâmetros da função esboçada na legenda.

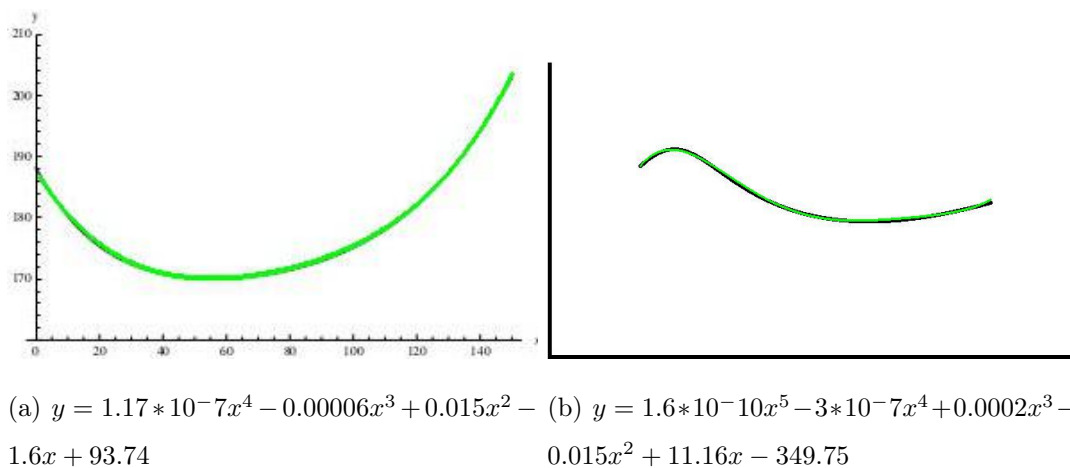
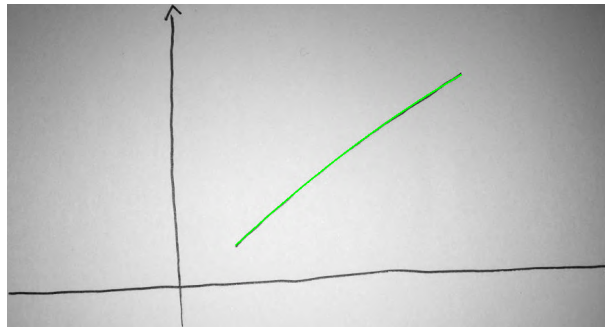
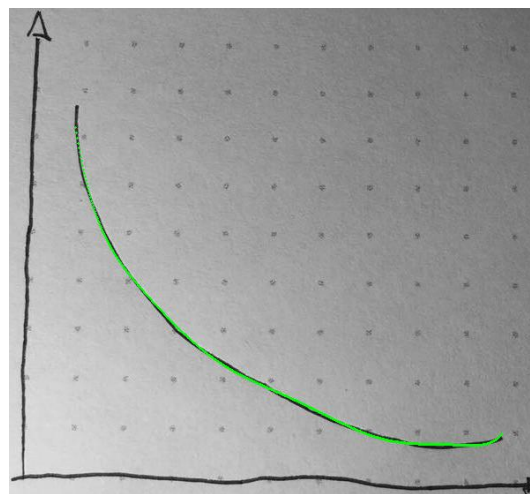


Figura 20: Resultados do ajuste polinomial

Apesar de todos os ajustes polinomiais ficarem próximos ou idênticos à curva de entrada, em alguns casos ocorreu *overfit*, especialmente nas curvas desenhadas à mão. A figura 21(a) mostra uma função que poderia ser bem aproximada por um polinômio de grau 1, no entanto, por se tratar de um desenho, apresentou uma leve concavidade que levou o ajuste à equação de segundo grau; a figura 21(b) mostra uma curva que obteve como resultado um polinômio de grau 8, apresentando diversas irregularidades e oscilações, neste caso uma função com um erro um pouco maior mas com menos irregularidades iria representar melhor a função.



$$(a) \ y = -0.00053x^2 + 1.01x - 21.52$$



$$(b) \ y = 4.16 * 10^{-17}x^8 - 1.06 * 10^{-13}x^7 + 1.14 * 10^{-10}x^6 - 6.71 * 10^{-8}x^5 + 0.00002x^4 - 0.005x^3 + 0.653x^2 - 47.35x + 1750.3$$

Figura 21: Ajustes insatisfatórios de funções polinomiais

4.3 AJUSTE EXPONENCIAL

O ajuste de todas as funções não lineares, inclusive da exponencial, foi feito de forma diferente do ajuste da função polinomial. O primeiro é determinar a função a ser minimizada; em seguida é utilizada uma rotina genérica de minimização provida pelo *scipy*. A rotina se mostrou bem flexível, provendo a implementação de diferentes opções de algoritmos de minimização. A fórmula usada para a minimização da exponencial foi

$$\sum_{i=1}^N [y_i - (e^{a*x_i+b} + c)]^2 \quad (4.1)$$

sendo (x_i, y_i) as coordenadas de cada ponto de entrada e **a**, **b**, **c** os parâmetros a serem determinados.

4.3.1 Método do Ajuste

A minimização da expressão 4.1 foi feita através da função *optimize.minimize* do *scipy*. A rotina *minimize* é adaptável, e permite a escolha de diversos métodos de minimização; a opção que mostrou melhor resultado para o ajuste da exponencial foi o método de Nelder-Mead. A minimização de Nelder-Mead mostrou bons resultados para várias classes de funções e foi escolhida como método para a maioria dos ajustes não-lineares feitos nesse projeto. Algumas características do método de Nelder-Mead fizeram dele especialmente apropriado para este caso de uso, como o fato de ser *derivative-free*, o que permite que o método funcione mesmo para entradas que apresentem muitos erros e funções a serem minimizadas que sejam não-diferenciáveis; o método também se mostrou não ser muito sensível a grandes discrepâncias nos valores dos parâmetros durante o ajuste. Uma descrição mais detalhada do algoritmo de Nelder-Mead pode ser encontrada no capítulo 2.

Além da função a ser minimizada, o algoritmo toma como entrada o vetor de parâmetros inicial a ser testado e a partir dele gera de forma automática o primeiro simples; a escolha do vetor inicial é importante pois afeta drasticamente o resultado final. O vetor de parâmetros inicial escolhido para a exponencial utilizou os valores $a = -0.9$, $b = -0.8$ e $c = 1$, os quais foram escolhidos após testes sequenciais comparando o erro após a execução do algoritmo com diferentes vetores de entrada.

4.3.2 Resultados

O resultado do ajuste da exponencial foi considerado bem satisfatório; em todos os casos em que a curva de entrada era uma exponencial ou próxima de uma o ajuste

apresentou erro baixo, com gráficos próximos da curva original. A figura 22 apresenta resultados do ajuste perfeito de duas funções exponenciais.

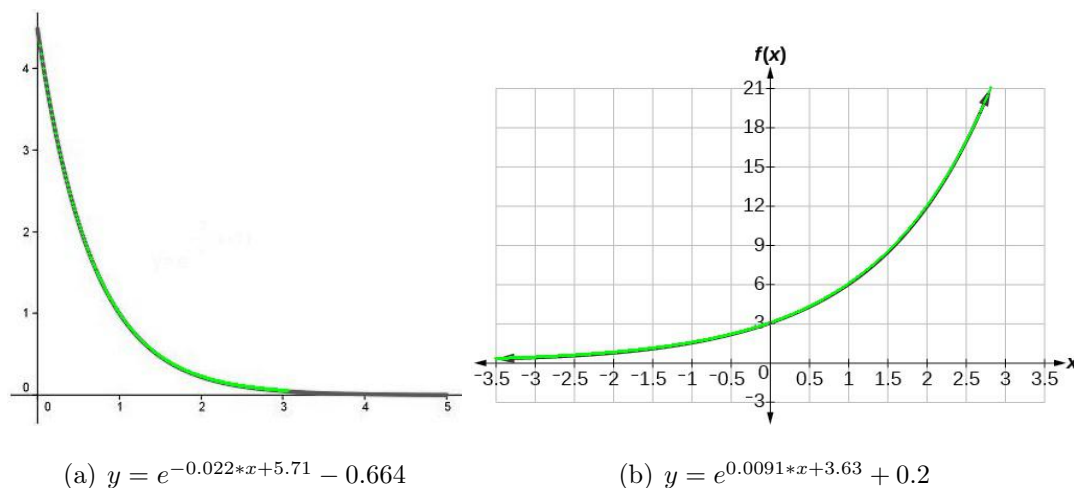


Figura 22: Ajuste perfeito de funções exponenciais

A figura 23 mostra resultados menos satisfatórios; a figura 23(a) apresenta uma função distante da curva de entrada; essa curva especificamente se mostrou de difícil ajuste para algumas das classes. O principal motivo é que, pela presença de números próximos ao eixo X, uma grande parte da função foi recortada nesta seção durante o processamento inicial para extração dos pontos; com isso algumas das funções convergiram para essa seção durante a minimização. A figura 23(b) teve um bom ajuste, porém o gráfico se desencontrou da curva original, próximo à região que apresentava irregularidade no desenho.

4.4 AJUSTE LOGARITMO

A próxima função ajustada foi a de logaritmo; o ajuste trouxe o obstáculo da impossibilidade do cálculo de logaritmo de valores negativos. A fórmula geral escolhida foi

$$a \log(x * b) + c$$

que não previne que valores negativos de x ou b causem erros no processamento, o que tornou impossível o ajuste de curvas que atravessam horizontalmente o eixo y.

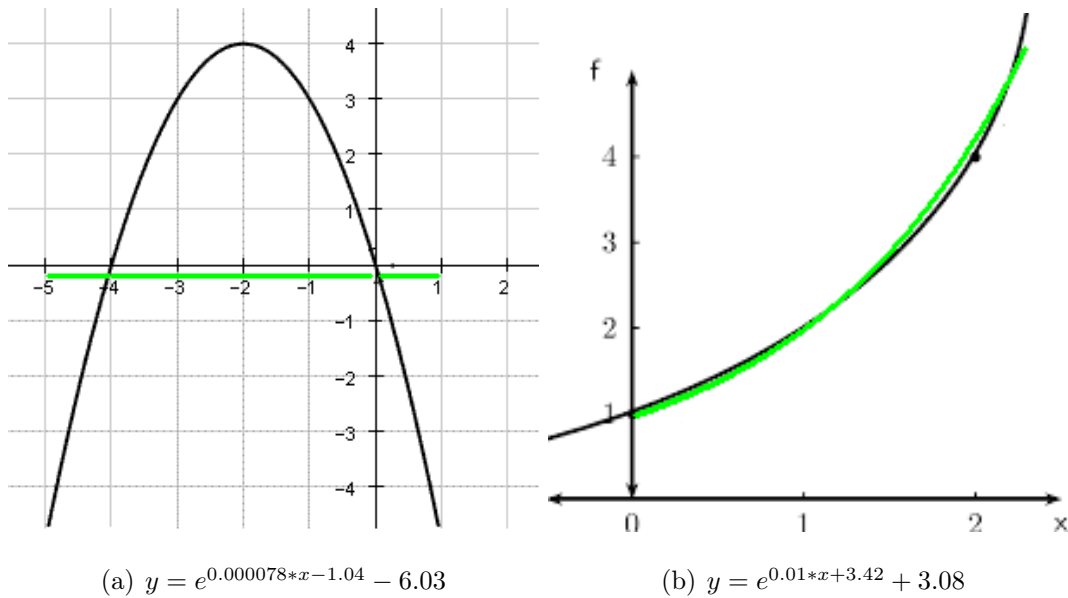


Figura 23: Ajustes insatisfatórios da classe exponencial

Para o ajuste da função logarítmica o objetivo é novamente minimizar o quadrado das diferenças com fórmula

$$\sum_{i=1}^N [y_i - (a \log x_i * b + c)]^2$$

o *log* utilizado foi o logaritmo natural, com base *e*.

4.4.1 Método do Ajuste

O ajuste foi feito com a rotina minimize, com o método de Nelder-Mead. Os valores de entrada para os parâmetros *a* e *c* foram determinados a partir do desvio padrão, levando em conta assim a amplitude da curva no eixo Y. O valor inicial do parâmetro *b* será discutido na seção 4.4.2.

4.4.2 Lidando com Valores de Logaritmo Inválidos

Por causa da limitação do logaritmo os valores tiveram que ser escolhidos cuidadosamente, visando evitar cálculos de logaritmos negativos. Primeiramente todas as

funções que apresentavam tanto valores negativos quanto positivos para x na mesma curva foram excluídas.

Depois foi feita a avaliação: para funções completamente positivas o valor de b foi inicializado como 1, garantindo uma convergência positiva. Quando a função era completamente negativa o valor de b escolhido foi -1. Isso permitiu englobar os dois grupos de curvas no ajuste sem causar erros.

4.4.3 Resultados

Para todas as imagens de teste que **eram** gráficos de funções logarítmicas os ajustes foram bons, exceto uma. Para as curvas que não eram gráficos logarítmicos se esperava que o erro fosse alto, neste caso os resultados variaram. A figura 24 mostra dois ajustes precisos de funções logarítmicas, uma negativa, a outra positiva.

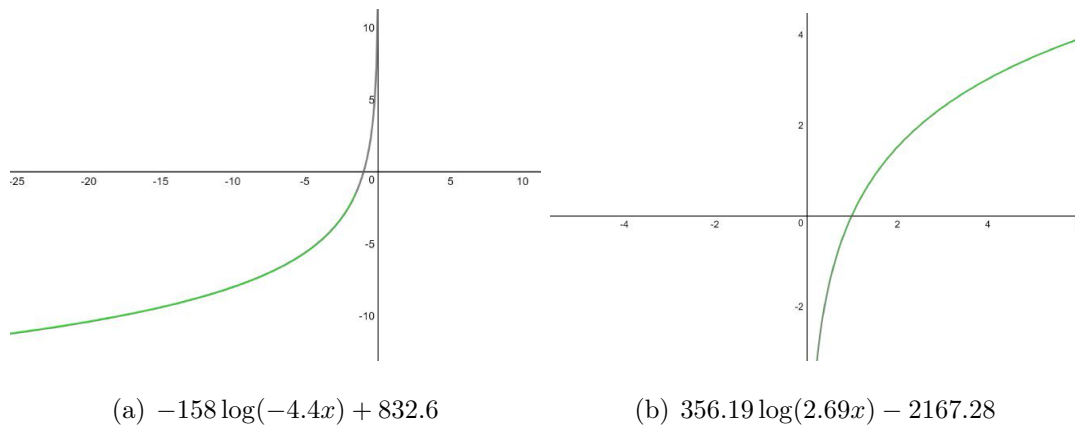


Figura 24: Ajuste perfeito de funções logarítmicas

A figura 25(b) mostra uma função não logarítmica, que apresenta um grande erro, apesar do gráfico de saída se manter na área da curva de entrada. O resultado mais preocupante foi o da figura 25(a), um gráfico de função logarítmica que não convergiu para a curva inicial, mesmo quando o ajuste foi alimentado com valores iniciais próximos à curva objetivo.

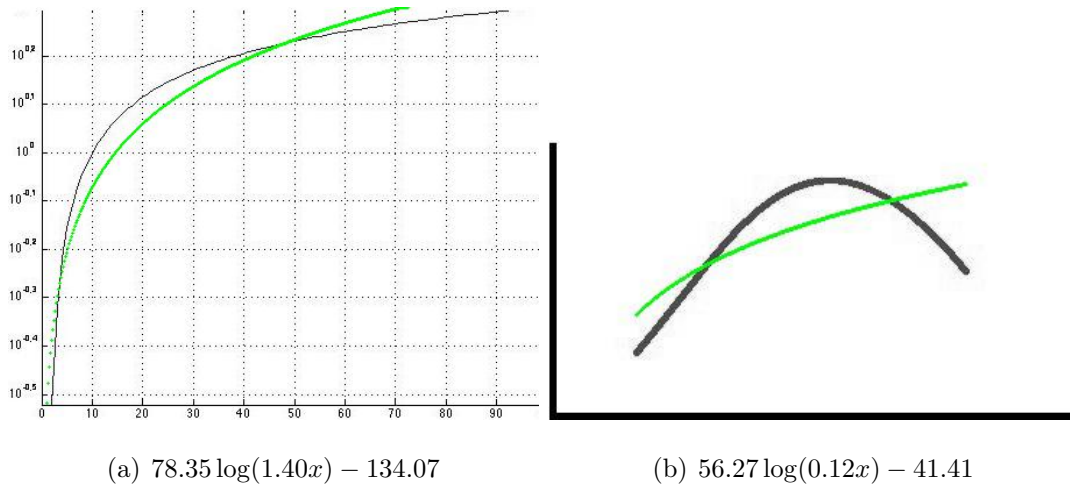


Figura 25: Ajuste insatisfatório de funções logarítmicas

4.5 AJUSTE GAUSSIANO

A gaussiana foi considerada uma função importante de se incluir no ajuste por sua relevância no campo da estatística. A fórmula escolhida foi a mais utilizada por estudantes e pesquisadores de estatística que possam se interessar em utilizar ou estender a ferramenta no futuro. Com isso a fórmula minimizada foi

$$\sum_{i=1}^N \left[y_i - \left(\frac{1}{a\sqrt{2\pi}} e^{-(x_i-b)^2/c^2} \right) \right]^2$$

O ajuste foi feito com o uso da rotina minimize, com o método de Nelder-Mead. A escolha dos parâmetros de ajuste iniciais foi feita se aproveitando das propriedades da gaussiana, com os valores iniciais de a e c iguais ao desvio padrão e o valor de b igual à média.

4.5.1 Resultados

Entre as figuras de teste apenas uma apresentava o gráfico de uma função gaussiana, que foi perfeitamente ajustada. Um dos gráficos da função racional apresentava o formato próximo ao clássico formato de “sino” da gaussiana; este também apresentou resultado próximo, apesar do ajuste não ser perfeito como esperado. Os dois

gráficos podem ser vistos na figura 26.

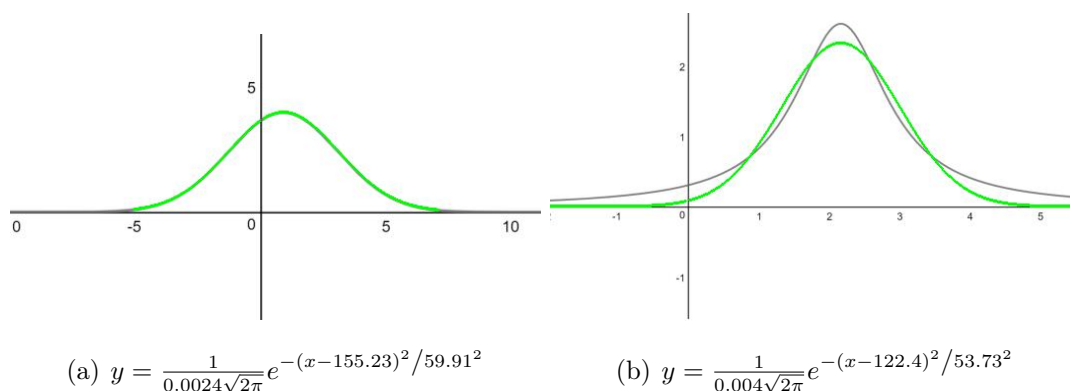


Figura 26: Resultados de ajuste gaussiano com baixo valor de erro

Como a função gaussiana apresenta pouca flexibilidade todas as curvas que não eram gaussianas apresentaram um grande erro, apesar da função apresentar crescimento e decrescimento proporcionais à taxa de variação da curva. A figura 27 ilustra dois destes casos.

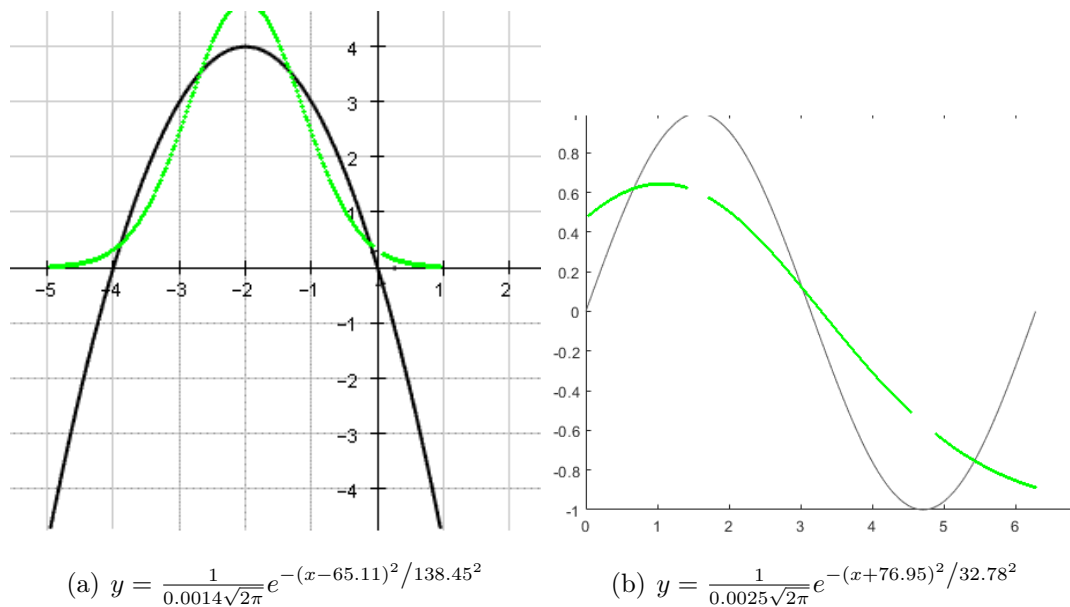


Figura 27: Ajustes da Gaussiana com altos valores de erro

4.6 AJUSTE FUNÇÃO RACIONAL

A próxima função ajustada foi a função racional, e a equação escolhida foi

$$\frac{1}{a * x^2 + b * x + c}$$

Como era importante a inclusão de pelo menos uma equação no formato fracionário no conjunto de funções ajustadas, se optou pela escolha de uma equação simples mas que abrange formatos bem variados de curvas. Com isso, a fórmula de cálculo do erro a ser minimizado foi

$$\sum_{i=1}^N \left[y_i - \frac{1}{a * x_i^2 + b * x_i + c} \right]^2$$

4.6.1 Método do Ajuste

A minimização foi feita com a rotina minimize com o método de Nelder-Mead, com os parâmetros de entrada $a = 0$, $b = -0.0001$ e $c = 0$. O melhor valor de entrada foi obtido após testes sucessivos, de maneira análoga à técnica para escolha de parâmetros da exponencial, descrita na seção 4.3.1.

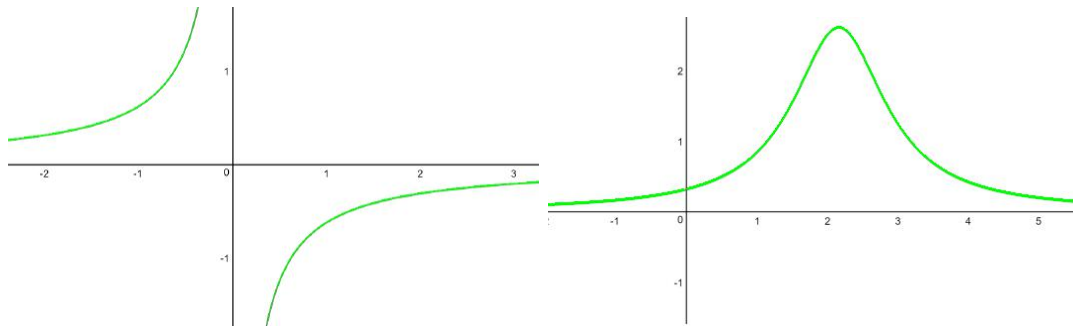
4.6.2 Resultados

O ajuste da função racional escolhida gerou gráficos em geral próximos das curvas de entrada, apresentando resultado perfeito nos casos em que a curva de entrada representa uma função racional. A figura 28 exhibe o ajuste de duas funções racionais, compostas da divisão de 1 por polinômios de diferentes graus.

Apesar de imperfeitos os ajustes de funções polinomiais também se mostraram próximos às curvas de entrada, no entanto, pela diferença de classe o somatório do erro ainda é alto (ver figura 29).

4.7 AJUSTE SENOIDE

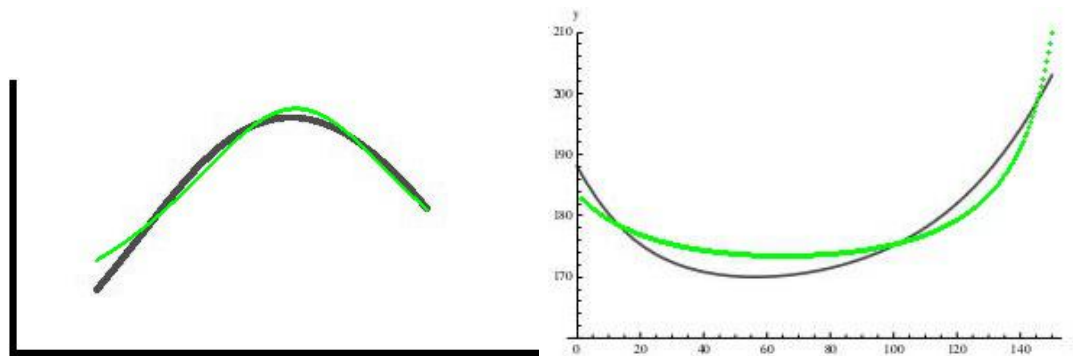
A função senoide se mostrou a mais desafiadora de se ajustar, se apresentando fortemente sensível aos valores iniciais para minimização, levando a uma implemen-



(a) $y = \frac{1}{2.47 \cdot 10^{-11} x^2 - 6.18 \cdot 10^{-5} x + 1.33 \cdot 10^{-5}}$

(b) $y = \frac{1}{1.65 \cdot 10^{-6} x^2 - 5.14 \cdot 10^{-4} x + 0.045}$

Figura 28: Ajuste perfeito da função racional



(a) $y = \frac{1}{-7.1 \cdot 10^{-7} x^2 + 1.63 \cdot 10^{-4} x + 0.012}$

(b) $y = \frac{1}{5.12 \cdot 10^{-7} x^2 - 2 \cdot 10^{-4} x + 0.025}$

Figura 29: Ajuste insatisfatório da função racional

tação diferente do ajuste em relação às outras funções não-lineares. A fórmula de ajuste escolhida foi

$$a * \sin(b * x + d) + c$$

sendo esta bem flexível, permitindo controlar amplitude, frequência angular, fase e altura. O erro a ser minimizado foi portanto

$$\sum_{i=1}^N [y_i - (a * \sin(b * x_i + d) + c)]^2$$

4.7.1 Método do Ajuste

O ajuste do seno foi feito com a função minimize aplicada sobre a fórmula do quadrado das diferenças, no entanto, diferentemente das outras funções não-lineares o método que apresentou melhor resposta foi o método de Powell.

Além disso o ajuste apresentou resultados ruins para parâmetros de entrada estáticos, levando a uma abordagem diferente na definição dos parâmetros iniciais, descrita em detalhe na seção 4.7.2.

4.7.2 Escolha de Valores Iniciais para o Algoritmo

Os parâmetros iniciais para a execução do método de Powell foram escolhidos inicialmente se baseando em características próprias dos dados de entrada. O ajuste de todos os gráficos respondeu bem a um valor de a baseado no desvio padrão, ao valor de d igual à média e a um valor de c inicializado com 0. No entanto os gráficos variaram muito na qualidade do ajuste com diferentes valores de b iniciais.

A solução foi tornar o valor de b adaptativo para cada um dos ajustes; o valor de b começa relativamente alto, a amplitude da curva no eixo x dividida por mil; com isso o período da função do seno fica bem curto. O resultado do ajuste com esse vetor inicial de valores é então avaliado: caso o erro continue muito alto é feito um novo ajuste, mas com o valor inicial de b mais próximo de 0. Isso é repetido até que se encontre um ajuste bom, ou até que b tenha um valor muito próximo de 0.

4.7.3 Resultados

Todas as curvas que expressavam perfeitamente a função seno tiveram um bom ajuste, mesmo com grande oscilação de altura e frequência. Dois exemplos podem ser observados na figura 30.

O resultado do ajuste aplicado sobre curvas representativas de polinômios foi menos satisfatório, apesar de determinadas curvas possuírem formatos que se encaixam em uma função seno, como os gráficos da figura 31. Durante a adaptação

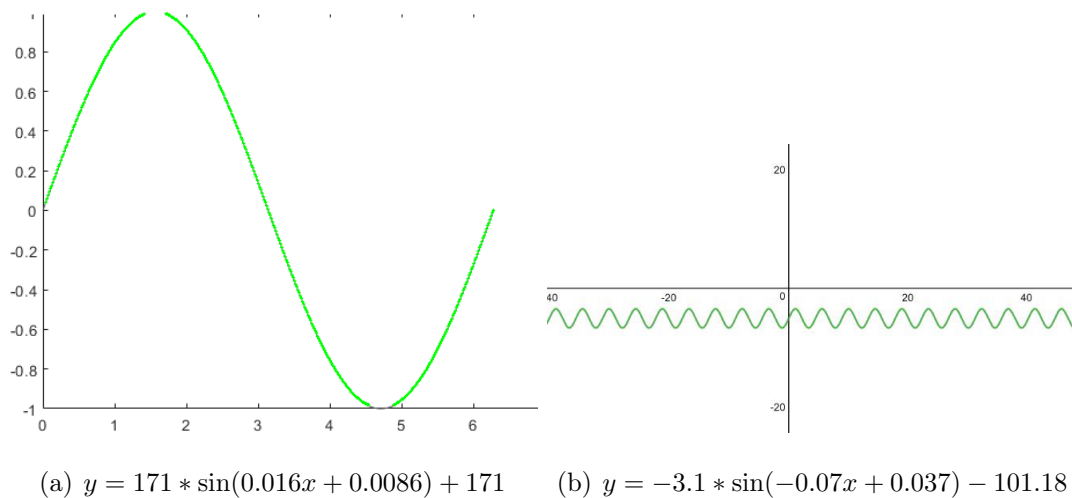


Figura 30: Ajuste perfeito da função seno

dos valores iniciais para a função seno foram priorizadas curvas que apresentassem senoides, especialmente aquelas que apresentassem a oscilação típica do seno. Com isso a definição de valores se tornou menos apropriada para curvas com período mais extenso.

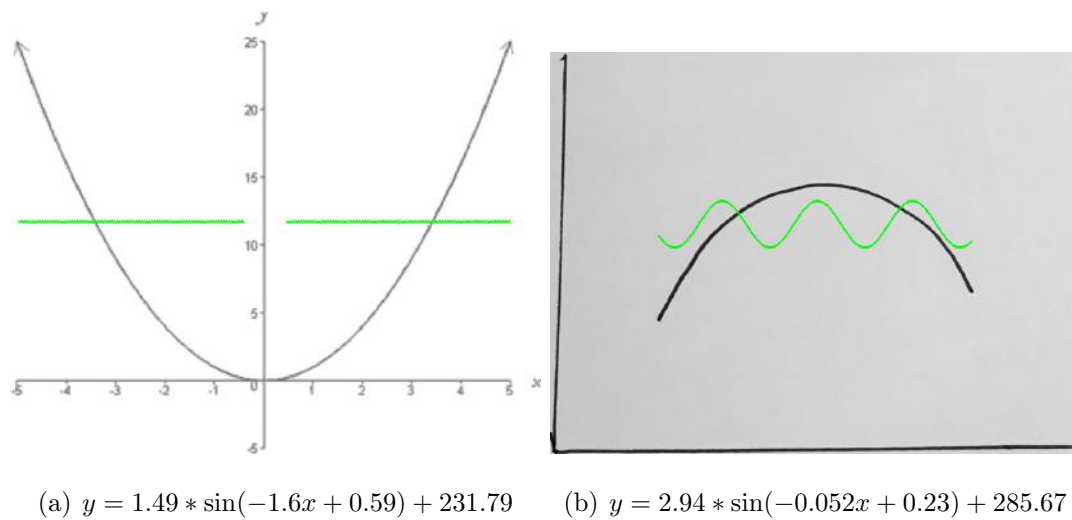


Figura 31: Ajustes insatisfatórios da função seno

4.8 AJUSTE FUNÇÃO MISTA

Após o ajuste satisfatório de todas as classes mencionadas nas seções anteriores foi implementado um ajuste final de uma função mista com equação composta pelo

somatório de todas as equações (no caso da função polinomial foi escolhido apenas um polinômio de grau 3 para entrar no somatório). Se trata então da minimização de uma equação de 19 parâmetros composta pela soma de cada equação, utilizando novamente o método dos mínimos quadrados.

4.8.1 Método do Ajuste

O método de ajuste utilizado foi a minimização de Nelder-Mead aplicada utilizando a rotina *minimize* e recebendo como vetor inicial de parâmetros um vetor feito pela junção de todos os parâmetros iniciais que foram bem sucedidos no ajuste de suas respectivas funções, com a exceção da função seno, que desta vez recebeu apenas parâmetros estáticos calculados a partir dos pontos de entrada.

4.8.2 Discussão dos Resultados

O ajuste da função mista não se mostrou bem sucedido, todos os ajustes apresentaram erros altos e nenhum deles representou bem a curva de entrada (Figura 32). Isso se deve a motivos variados; ao adicionar tantos parâmetros e diferentes equações muito distintas ao problema de minimização ele se torna muito mais complexo ¹, extremamente sensível à escolha dos parâmetros iniciais e método de minimização.

Um ajuste bem sucedido da função mista dependeria de uma escolha cuidadosa de tais fatores, neste projeto o foco principal foi realizar um bom ajuste para cada uma das classes separadamente.

4.9 AJUSTE GLOBAL

Após a obtenção de um resultado satisfatório em cada uma das funções isoladas, começa o processo de buscar pelo ajuste mais adequado para cada curva de entrada. O processo ocorre a partir da execução do ajuste para cada uma das classes de

¹Parte da sequência de pontos pode ser bem representada por uma classe de funções, enquanto outra parte pode ser representada por outra classe

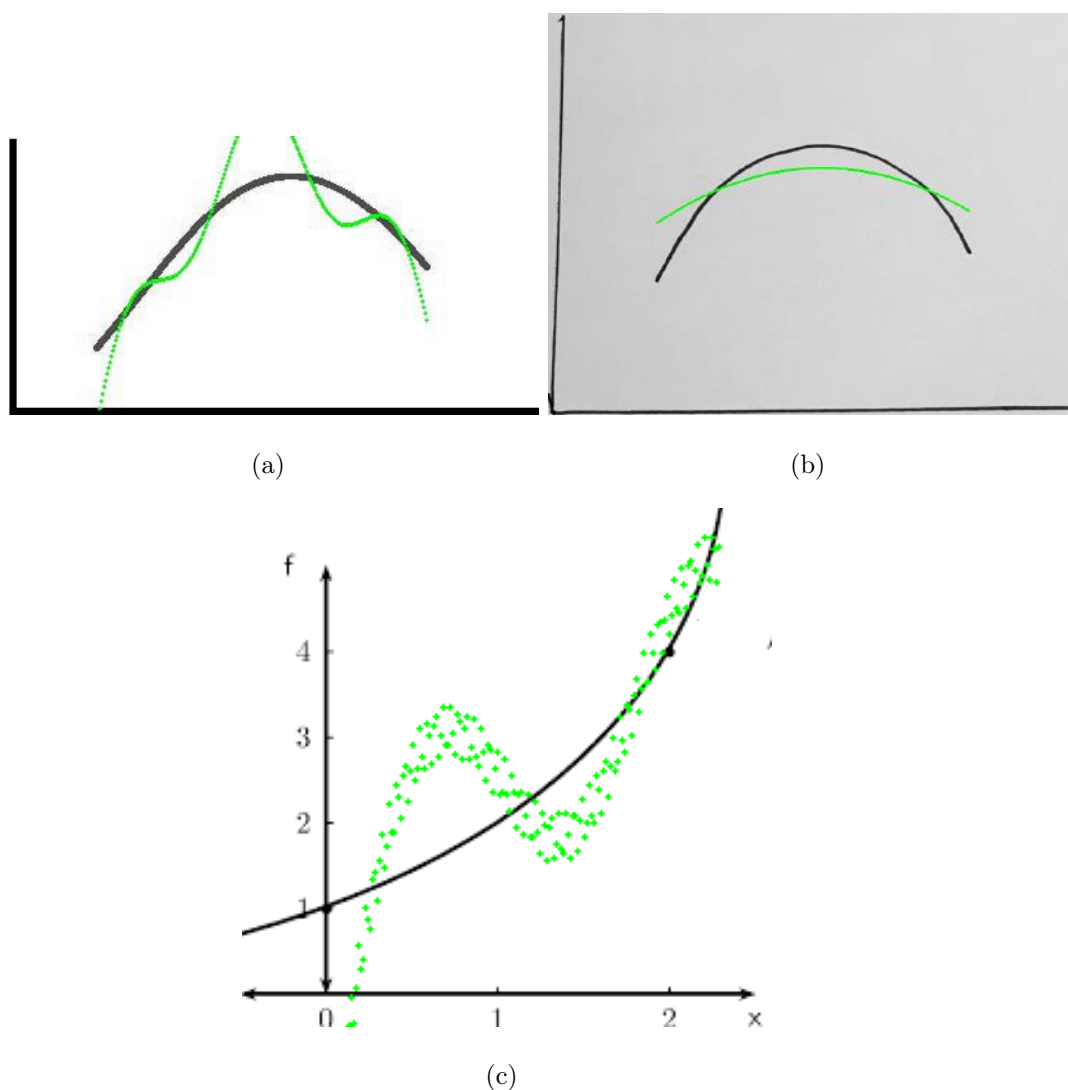


Figura 32: Ajustes da função Mista

função seguida por uma comparação dos resultados.

A saída é a classe de função que se mostrou mais apropriada, os parâmetros da equação que proveram o melhor ajuste e o esboço da função.

4.9.1 Método para a Escolha

Se assume no primeiro momento que a classe de função que oferece o menor erro possível é a mais apropriada. Cada uma das classes, polinomial, exponencial, senoide, racional, gaussiana e logarítmica, é ajustada e se recebe como resposta o erro do ajuste feito por aquela classe (a soma do quadrado das diferenças). O erro

retornado por cada uma das chamadas é avaliado, para eleger a melhor candidata para a função.

O melhor erro encontrado é inicializado como infinito; a cada iteração em que calculamos um novo ajuste e obtemos um novo erro comparamos ele com o erro atual. Se a nova classe mostrar um resultado melhor, passa a ser a nova candidata de melhor classe de função e seu erro se torna o erro mínimo. A avaliação de se a classe testada será a nova candidata não é baseada apenas no valor do erro, mas também na complexidade da função, tal avaliação será descrita na seção 4.9.2.

4.9.2 Método de Prevenção de *Overfitting*

Para prevenção do *overfitting* a complexidade da função é levada em conta no momento de comparação da atual melhor classe e ajuste com cada nova classe testada. Para isso a comparação foi feita em duas partes, sempre que era obtida um função com resultado de erro menor do que o atual erro mínimo era feito um segundo teste.

Foi estabelecida uma ordem de hierarquia de complexidade, atribuindo como um fator de complexidade um valor numérico para cada uma das classes (tabela 1). Ao ser encontrada uma classe com erro menor do que a classe atual a operação é:

- Se a classe possui erro inferior e complexidade igual ou inferior à da classe atual então ela se torna a nova candidata a melhor ajuste
- Se a classe possui erro inferior e a complexidade é superior à da classe atual se avalia a diferença entre os erros. Se a diferença for pouco significativa se mantém a classe atual, caso contrário a classe atual é substituída.

Essa avaliação se mostrou especialmente importante quando o ajuste polinomial retornou um polinômio de grau 8, podendo tomar o lugar de classes mais óbvias em determinadas curvas por causa de pequenos erros nos dados de entrada devido à qualidade do traçado da curva.

Polinomial Grau 0 - 5	1
Gaussiana	4
Racional	3
Senoide	4
Exponencial	2
Logarítmica	3
Polinomial Grau 6	4
Polinomial Grau 7 - 8	5

Tabela 1: Ordem de complexidade estabelecida para cada classe

Foi feita uma tentativa de amostragem dos pontos para reduzir o *overfitting*, onde o processo de ajuste foi feito com apenas um terço dos pontos de entrada, distribuídos de forma regular. Após o ajuste algumas das curvas que apresentavam ajustes bons tiveram o ajuste afetado negativamente pelo *sampling* (amostragem) dos pontos, enquanto as curvas que antes apresentavam *overfitting* continuaram apresentando ajustes inadequados. Os resultados obtidos não foram superiores aqueles obtidos com a utilização de todos os pontos, o que não motivou o desenvolvimento da pesquisa nessa direção; porém, isso não descarta um estudo mais detalhado no futuro sobre o assunto pois o emprego de estratégias adequadas para a escolha dos pontos ainda pode ser promissor.

4.9.3 Discussão dos Resultados

Na tabela 2 se encontra uma listagem de todas as curvas ajustadas, especificando a classe de função esperada como resultado (somente no caso em que a imagem original foi baseada em um gráfico computadorizado), a classe de curva selecionada como a melhor pelo algoritmo e a fórmula da função ajustada. Para cada uma das curvas ajustadas também foi calculado o erro médio por ponto de entrada, ou seja, a diferença média entre os pontos de entrada e o gráfico da função de saída.

Pode ser observado que a maioria dos ajustes apresentou erro médio baixo, abaixo de 10. E a esmagadora maioria apresentou erro médio abaixo de 50, um bom resul-

tado de ajuste (considerando que cada unidade é o tamanho de um pixel).

As linhas da tabela marcadas em cinza representam as imagens de entrada que não tiveram bons resultados na etapa de pré-processamento. Tais figuras ou não puderam ser ajustadas ou apresentaram ajustes péssimos por interferência de diversas linhas sobre a curva, que não foram limpas na primeira etapa.

Em geral os resultados das imagens que foram pré-processadas com sucesso foi bom, apresentando erro baixo e com classes de resposta sensatas. Houve um aumento do erro em casos em que o ajuste da classe esperada não ficou tão bom, tais casos foram discutidos ao longo deste capítulo. No entanto, por vezes, ocorreu um aumento do erro simplesmente porque a imagem original ainda apresentava ruído, como pequenos traços que atravassem a curva.

	Classe Esperada	Classe do Ajuste	Equação do Ajuste	Erro Médio por Ponto
Figura 33	-	Polinômio de Grau 5	$y = 1.6 * 10^{-10}x^5 - 3 * 10^{-7}x^4 + 0.0002x^3 - 0.015x^2 + 11.16x - 349.75$	1.3307
Figura 34	-	Polinômio de Grau 4	$y = 3.26 * 10^{-8}x^4 + 0.00004x^3 - 0.025x^2 + 6.9x - 406.75$	1.7543
Figura 35	-	Seno	$y = -97.8 * \sin(0.013x + 2.14) - 65.22$	0.4758
Figura 36	Polinômio	Polinômio de Grau 2	$y = 0.0038x^2 + 0.0063x - 0.644$	0.9818
Figura 37	-	Polinômio de Grau 4	$y = -4.32 * 10^{-8}x^4 - 3.7 * 10^{-6}x^3 - 0.0025x^2 - 0.06x + 82.84$	1.0995
Figura 38	-	Polinômio de Grau 2	$y = 0.0036x^2 - 0.36x + 63.42$	8.4949
Figura 39	-	Polinômio de Grau 2	$y = -0.00053x^2 + 1.01x - 21.52$	0.6576
Figura 40	Exponencial	Exponencial	$y = e^{-0.022*x+5.71} - 0.664$	0.9430
Figura 41	Exponencial	Polinômio de Grau 4	$y = 5.69 * 10^{-7}x^4 - 0.00014x^3 + 0.015x^2 - 0.076x + 40$	2.3835
Figura 42	Polinômio	Polinômio de Grau 4	$y = 5.69 * 10^{-7}x^4 - 0.00014x^3 + 0.015x^2 - 0.076x + 40$	0.3816
Figura 43	Seno	Seno	$y = -136.62 * \sin(-0.041x - 0.033) - 0.36$	1.7930
Figura 44	Seno	Seno	$y = -3.1 * \sin(-0.07x + 0.037) - 101.18$	0.5261
Figura 45	Seno	Seno	$y = 171 * \sin(0.016x + 0.0086) + 171$	0.4289
Figura 46	Seno	Seno	$y = 171 * \sin(0.016x + 0.0086) + 171$	0.7620
Figura 47	Logaritmo	Polinômio de Grau 8	$y = -3.07 * 10^{-16}x^8 + 5.4 * 10^{-13}x^7 - 3.97 * 10^{-10}x^6 + 1.57 * 10^{-7}x^5 - 0.00003x^4 + 0.005x^3 - 0.409x^2 + 18.1x - 98.13$	64.9626
Figura 18	Logaritmo	Sem ajuste	-	-
Figura 48	Logaritmo	Logaritmo	$y = -158 \log(-4.4x) + 832.6$	0.5356
Figura 49	Logaritmo	Logaritmo	$y = 356.19 \log(2.69x) - 2167.28$	2.4348
Figura 50	Gaussiana	Gaussiana	$y = \frac{1}{0.0024\sqrt{2\pi}} e^{\frac{(x-155.23)^2}{59.91^2}}$	0.1751
Figura 51	Racional	Racional	$y = \frac{1}{2.47*10^{-11}x^2-6.18*10^{-5}x+1.33*10^{-5}}$	0.3589
Figura 52	Racional	Racional	$y = \frac{1}{1.65*10^{-6}x^2-5.14*10^{-4}x+0.045}$	0.1709
Figura 53	-	Polinômio de Grau 5	$y = 2.3 * 10^{-11}x^5 - 9.7 * 10^{-9}x^4 - 3.4 * 10^{-6}x^3 + 0.0033x^2 - 0.98x + 102.29$	5.5284
Figura 54	-	Polinômio de Grau 8	$y = -3.07 * 10^{-16}x^8 + 5.4 * 10^{-13}x^7 - 3.97 * 10^{-10}x^6 + 1.57 * 10^{-7}x^5 - 0.00003x^4 + 0.005x^3 - 0.409x^2 + 18.1x - 98.13$	28.0296
Figura 55	Polinômio	Polinômio de Grau 1	$y = 2.99x + 134.67$	1.2677
Figura 56	Polinômio	Polinômio de Grau 2	$y = -0.021x^2 - 3.37x - 1.65$	1.8923
Figura 57	Polinômio	Polinômio de Grau 2	$y = 0.021x^2 - 3.02x - 185$	14.7289
Figura 58	Exponencial	Exponencial	$y = e^{0.0091*x+3.63} + 0.2$	2.6650
Figura 59	Exponencial	Exponencial	$y = e^{0.051*x+2.44} - 1.19$	4.0624
Figura 60	Polinômio	Polinômio de Grau 7	$y = -6.7 * 10^{-14}x^7 + 1.82 * 10^{-11}x^6 + 6.37 * 10^{-10}x^5 - 3.77 * 10^{-7}x^4 + 0.000011x^3 + 0.0008x^2 + 0.039x + 65.16$	13741.4
Figura 61	Polinômio	Polinômio de Grau 7	$y = 1.47 * 10^{-14}x^7 - 9.27 * 10^{-12}x^6 + 1.43 * 10^{-9}x^5 + 9.35 * 10^{-8}x^4 - 0.000028x^3 - 0.00004x^2 + 0.14x + 39.76$	9826.5
Figura 62	-	Polinômio de Grau 7	$y = 1.65 * 10^{-14}x^7 - 1.66 * 10^{-11}x^6 + 5.26 * 10^{-9}x^5 - 2.79 * 10^{-7}x^4 - 0.00012x^3 + 0.02x^2 - 1.33x - 391.66$	6986.2
Figura 63	-	Polinômio de Grau 8	$y = -5.26 * 10^{-18}x^8 + 4.53 * 10^{-15}x^7 - 8.84 * 10^{-14}x^6 - 7.45 * 10^{-10}x^5 + 1.09 * 10^{-7}x^4 + 0.00003x^3 - 0.005x^2 - 0.77x + 220.69$	11911.7

Tabela 2: Resultados Ajuste Global

5 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho consistiu na implementação de um sistema de tratamento de imagem, extração de características e aproximação de funções.

A estratégia de busca pela coordenada dos pontos que formam a curva presente na imagem, realizada no capítulo 3, fez com que a maioria das imagens de teste tenha sido preparada com sucesso, se mostrando totalmente apropriada para a etapa de processamento dos dados, com exceção de todas as imagens que possuem gradeado desenhado ao longo do desenho, com a mesma intensidade dos outros elementos e de uma das imagens de teste; nesses casos o algoritmo não foi capaz de achar um eixo X e Y, impossibilitando a etapa de processamento. A fase de pré-processamento exigiu a pesquisa e implementação de algoritmos conhecidos assim como a elaboração e adaptação de alguns algoritmos próprios.

A etapa de ajuste teve resultados satisfatórios, apresentando baixo erro em todas as curvas que haviam reagido bem ao pré-processamento, se mostrando apropriada para aproximação de problemas simples. O ajuste foi fortemente baseado na teoria de álgebra linear e cálculo numérico, e se tratou de um extenso processo de tentativa e erro.

O código com a implementação de toda a ferramenta será disponibilizado publicamente na *internet* para uso e expansão, com a licença de uso **GNU GPLv3**.

Para trabalhos futuros se sugere uma expansão da base de imagens de teste, possibilitando uma melhor análise dos resultados e evitando a adaptação exagerada a esse grupo de imagens. Na etapa de pré-processamento seria muito interessante introduzir uma técnica para remoção de linhas paralelas repetitivas atravessando a curva, muito importante para ajuste de funções esboçadas sobre um fundo quadriculado, também é uma melhoria importante o processo de refinação da detecção de eixos, diminuindo o número de casos de eixos não localizados.

Os resultados inferiores do ajuste das funções logarítmicas em específico torna interessante a busca por equações de ajuste mais apropriadas para tal classe. Uma

sugestão de equação de otimização promissora para a função logarítmica é a *função de erro qui-quadrado* [4]

$$\sum_{i=1}^N \frac{[\phi(x_i) - y_i]^2}{\sigma_{y_i}^2} \quad (5.1)$$

onde ϕ é a equação geral da curva a ser ajustada. (x_i, y_i) as coordenadas de cada ponto de entrada e $\sigma_{y_i}^2$ os valores das variâncias (uma para cada y_i).

Outra expansão de muita aplicação seria adicionar a identificação de escala no gráfico; isso seria um processo bem mais extenso, exigindo a identificação de marcações nos eixos e identificação de números. Por fim, a etapa de ajuste pode ser melhorada com a correção do ajuste da função mista, buscando um algoritmo de minimização e valores de entrada mais adequados e também com a adição de ainda mais classes de função ao ajuste.

Diferentes seções deste trabalho podem ser utilizadas como referência em implementações futuras para projetos em processamento de imagem e álgebra linear.

REFERÊNCIAS

- [1] ACQUAH, C., DATSKOV, I., MAWARDI, A., ZHANG, F., ACHENIE, L., PITCHUMANI, R., E SANTOS, E. Optimization under uncertainty of a composite fabrication process using a deterministic one-stage approach. *Computers and Chemical Engineering* (2005), 947–960.
- [2] Skeletonize documentation. http://scikit-image.org/docs/dev/auto_examples/edges/plot_skeleton.html. Acessado: 05-09-2018.
- [3] Image analysis: Morphological operations. <https://sites.ualberta.ca/~ccwj/teaching/image/morph/>. Acessado: 01-07-2018.
- [4] DRAPER, N. R., E SMITH, H. *Applied Regression Analysis*, 3 ed. John Wiley, 1998.
- [5] Engauge digitizer. <http://markummitchell.github.io/engauge-digitizer/>. Acessado: 01-12-2018.
- [6] FELLER, W. *An Introduction to Probability Theory and Its Applications*, 2 ed. John Wiley and Sons, 1971.
- [7] Getdata graph digitizer. <http://getdata-graph-digitizer.com/>. Acessado: 01-12-2018.
- [8] HE, L., REN, X., GAO, Q., ZHAO, X., YAO, B., E CHAO, Y. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition* 70 (2017), 25–43.
- [9] LIN, X., E OTOBE, K. Hough transform algorithm for real-time pattern recognition using an artificial retina camera. *Opt. Express* 8, 9 (2001), 503–508.
- [10] LYON, R. F. A brief history of ‘pixel’. *Digital Photography II, IS&T/SPIE Symposium on Electronic Imaging* (2006).
- [11] MATHEWS, J. H., E FINK, K. K. *Numerical Methods Using Matlab*, 4 ed. Prentice-Hall Inc., 2004.

- [12] Opencv - open source computer vision library. <https://opencv.org/>. Acesso: 01-07-2018.
- [13] PRESS, W., FLANNERY, B., TEUKOLSKY, S., E VETTERLING, W. *Numerical Recipes in C: the Art of Scientific Computing*, 2 ed. Cambridge University Press, 1988.
- [14] Scikit image - image processing in python. <http://scikit-image.org/>. Acesso: 01-07-2018.
- [15] SILVA, E. C. N. Apostila de otimização aplicada ao projeto de sistemas mecânicos, 2018.
- [16] TOMASI, C., E MANDUCHI, R. Bilateral filtering for gray and color images. *IEEE International Conference on Computer Vision* (1998).
- [17] TORBERT, S. Applied computer science. *Springer 2* (2016).

ANEXOS

Aqui estão listadas todas as figuras de teste com os resultados do ajuste global de cada uma. Cada imagem apresenta o gráfico da função ajustada esboçado em verde sobre a imagem inicial. As imagens originais foram tratadas, sendo colocadas em escala de cinza e recortadas, para melhor visualização, ou seja, a imagem original em geral era maior, com a parte desenhada posicionada em regiões diversas da imagem digital. A fórmula da função final de cada curva se encontra na legenda de cada figura.

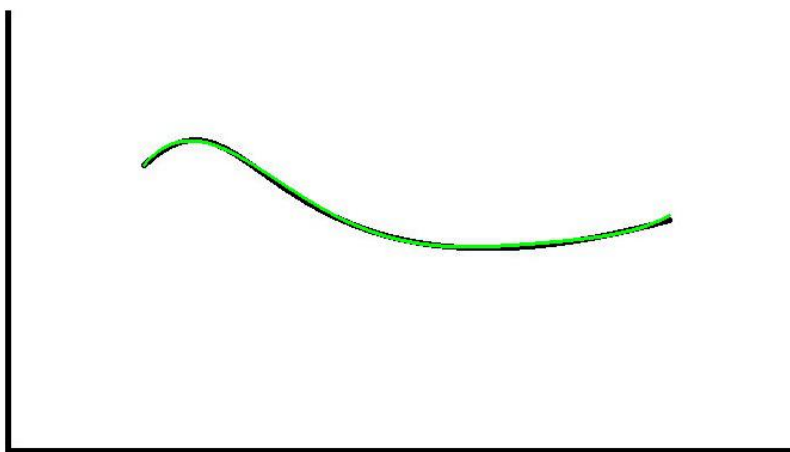


Figura 33: Ajuste com polinômio: $y = 1.6 * 10^{-10}x^5 - 3 * 10^{-7}x^4 + 0.0002x^3 - 0.015x^2 + 11.16x - 349.75$

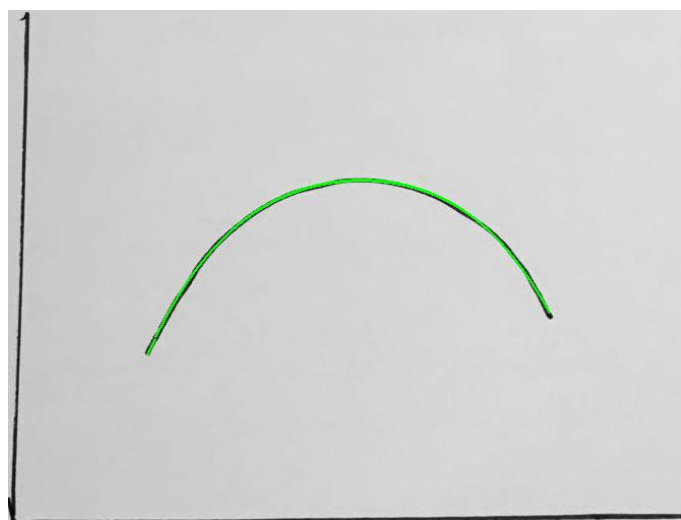


Figura 34: Ajuste com polinômio: $y = 3.26 * 10^{-8}x^4 + 0.00004x^3 - 0.025x^2 + 6.9x - 406.75$

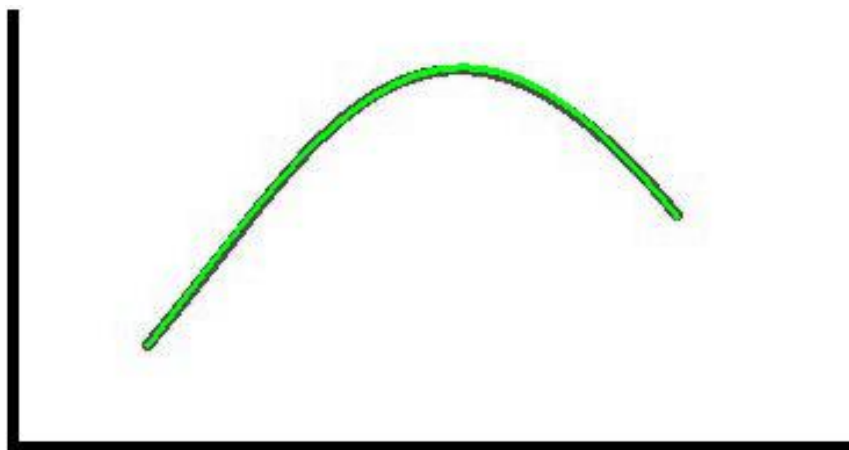


Figura 35: Ajuste com seno: $y = -97.8 * \sin(0.013x + 2.14) - 65.22$

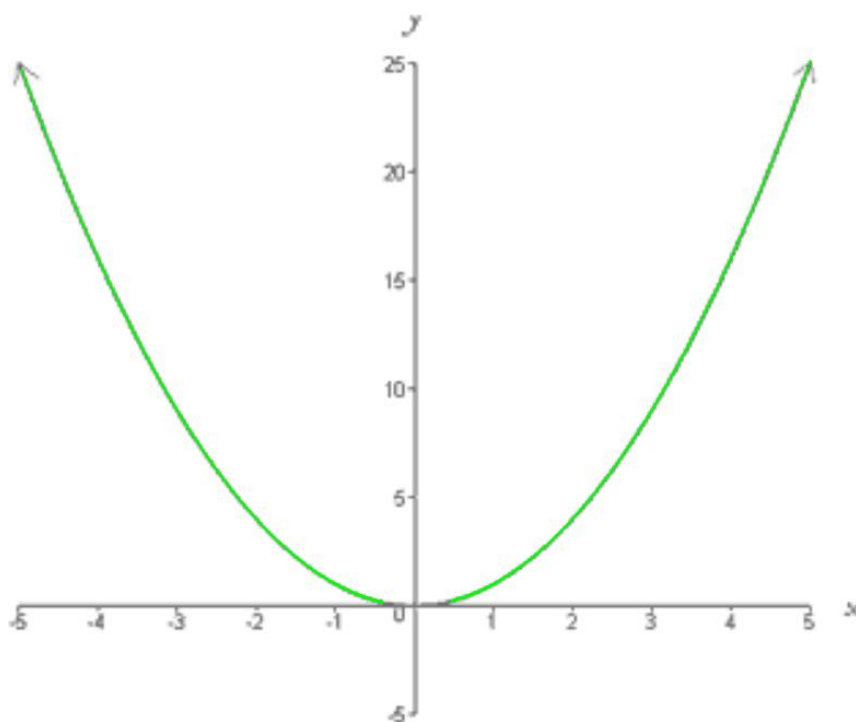


Figura 36: Ajuste com polinômio: $y = 0.0038x^2 + 0.0063x - 0.644$

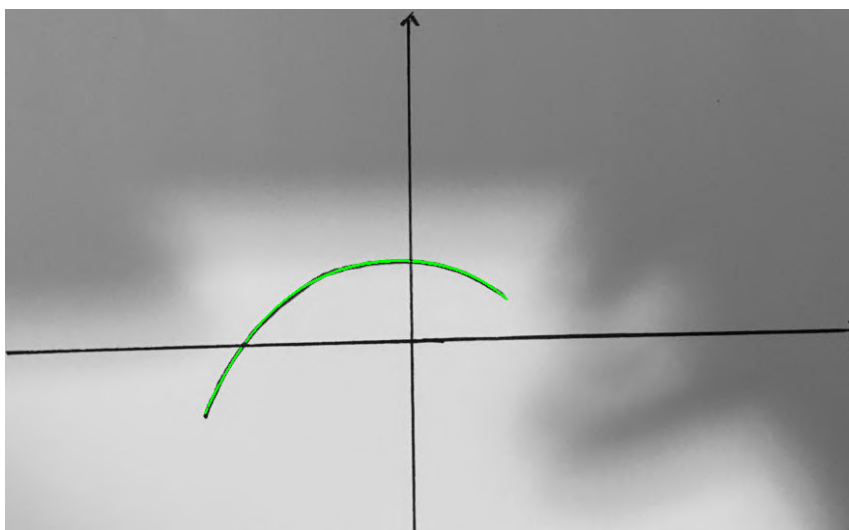


Figura 37: Ajuste com polinômio: $y = -4.32 * 10^{-8}x^4 - 3.7 * 10^{-6}x^3 - 0.0025x^2 - 0.06x + 82.84$

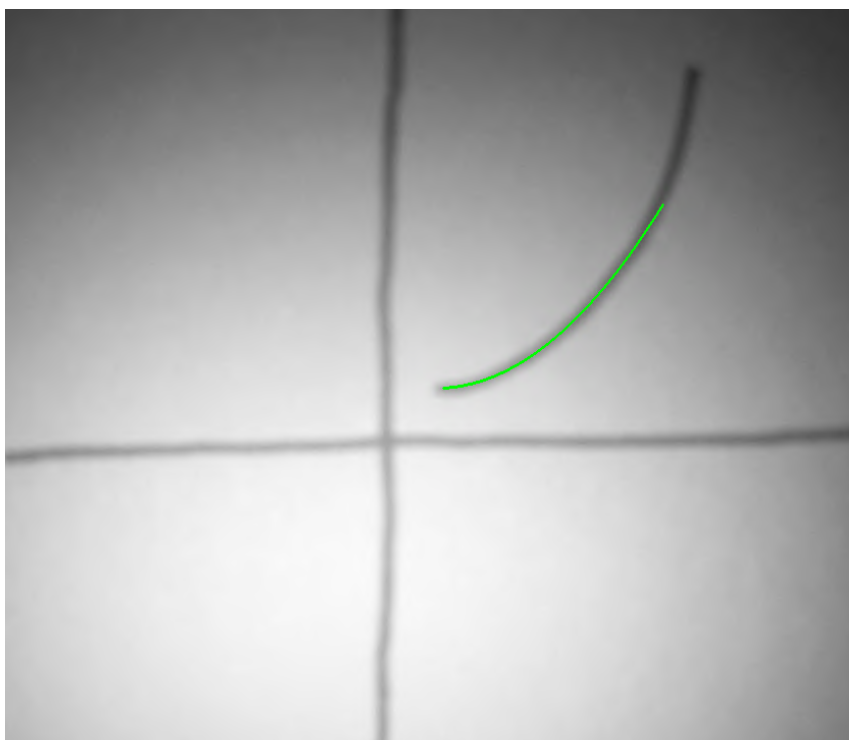


Figura 38: Ajuste com polinômio: $y = 0.0036x^2 - 0.36x + 63.42$



Figura 39: Ajuste com polinômio: $y = -0.00053x^2 + 1.01x - 21.52$

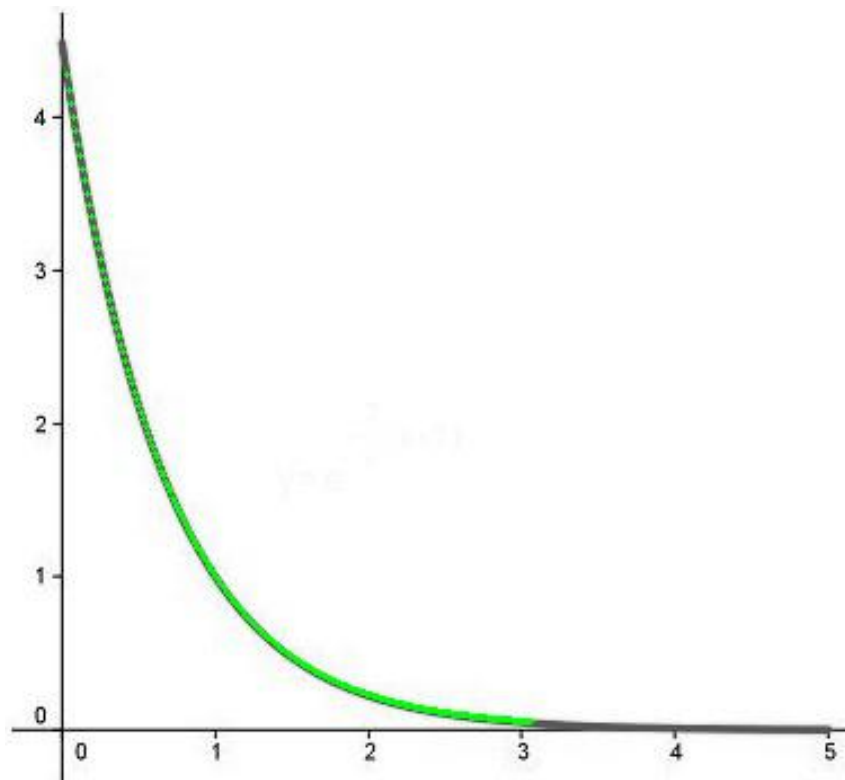


Figura 40: Ajuste com exponencial: $y = e^{-0.022*x+5.71} - 0.664$

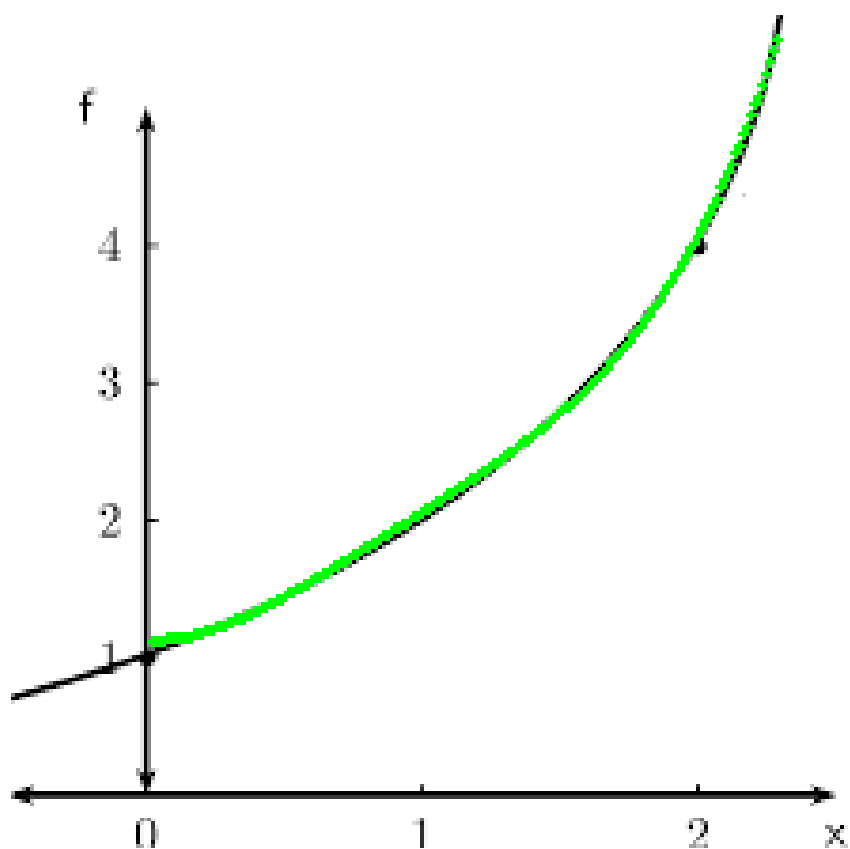


Figura 41: Ajuste com polinômio: $y = 5.69 \cdot 10^{-7}x^4 - 0.00014x^3 + 0.015x^2 - 0.076x + 40$

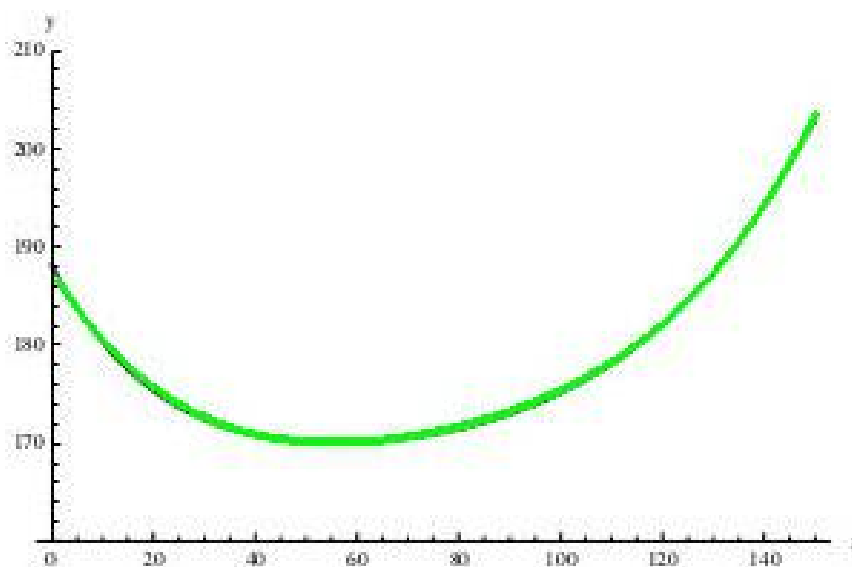


Figura 42: Ajuste com polinômio: $y = 1.17 \cdot 10^{-7}x^4 - 0.00006x^3 + 0.015x^2 - 1.6x + 93.74$

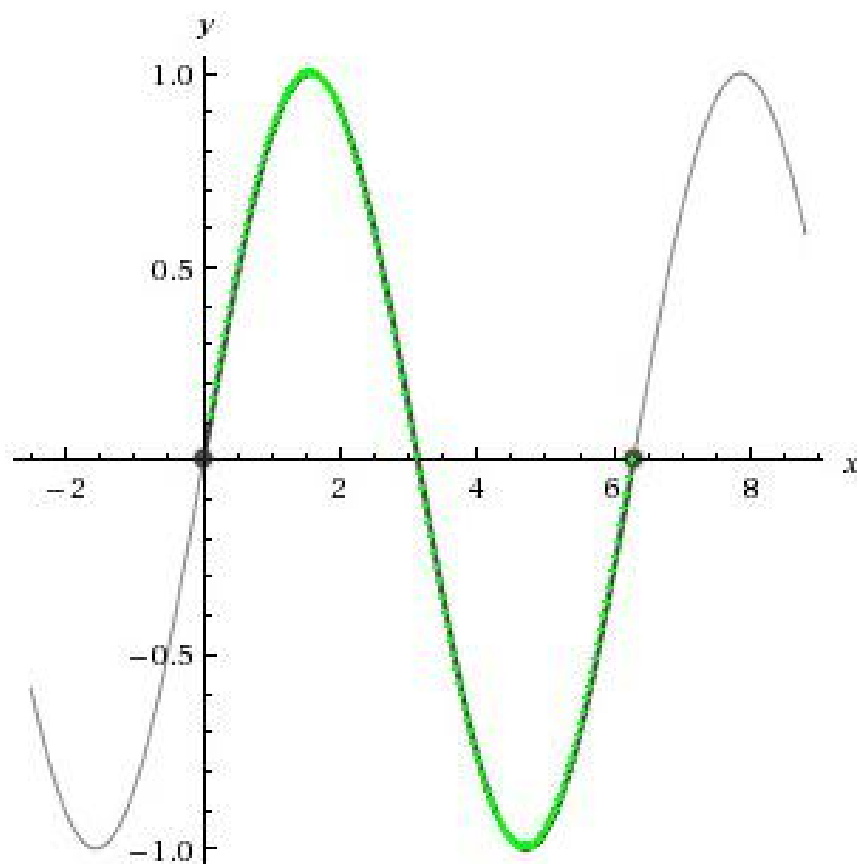


Figura 43: Ajuste com seno: $y = -136.62 * \sin(-0.041x - 0.033) - 0.36$

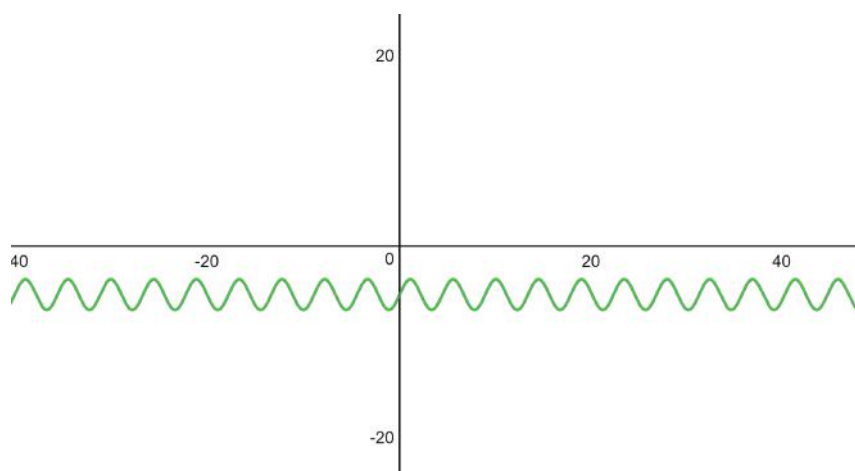


Figura 44: Ajuste com seno: $y = -3.1 * \sin(-0.07x + 0.037) - 101.18$

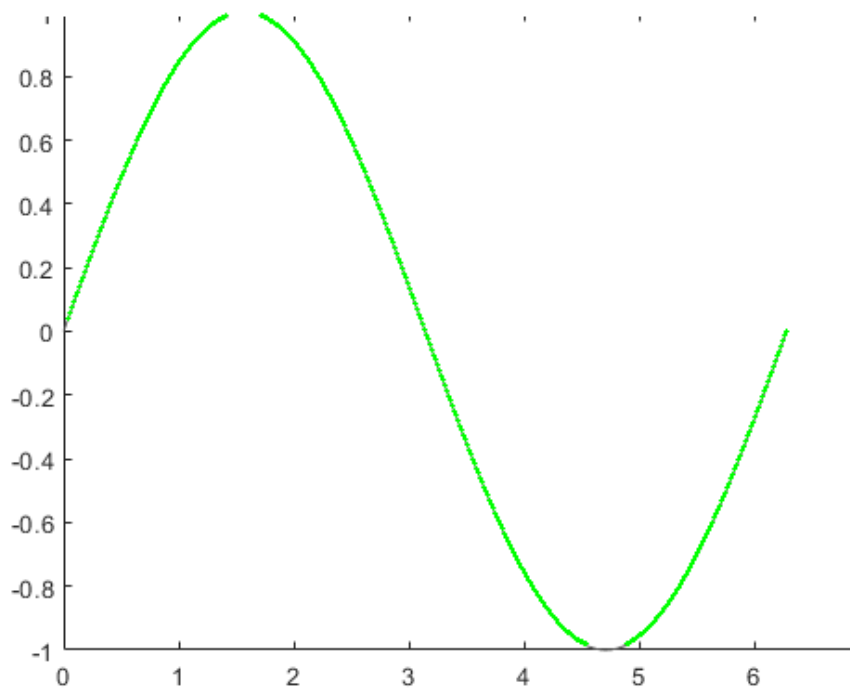


Figura 45: Ajuste com seno: $y = 171 * \sin(0.016x + 0.0086) + 171$

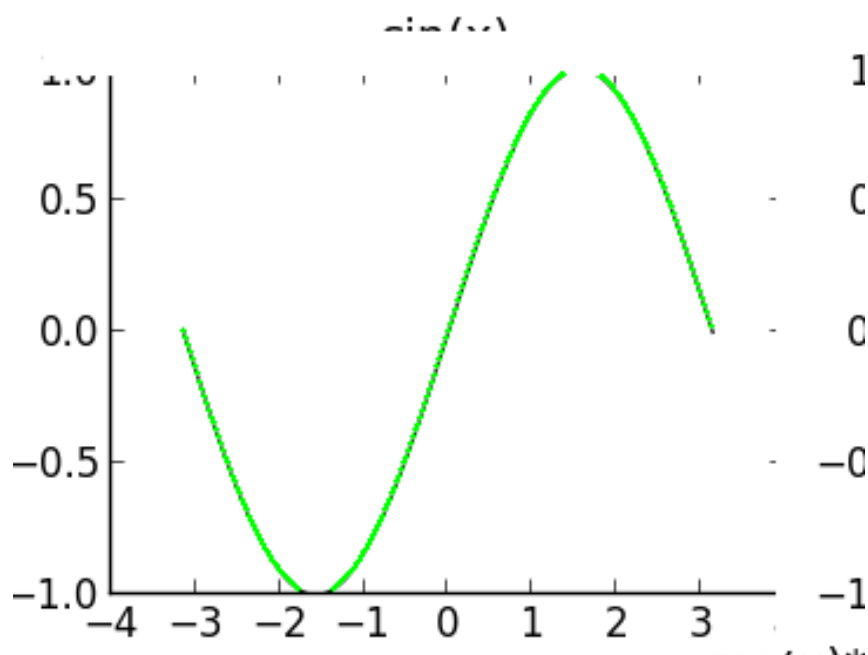


Figura 46: Ajuste com seno: $y = -111.85 * \sin(0.027x - 0.86) + 111.25$

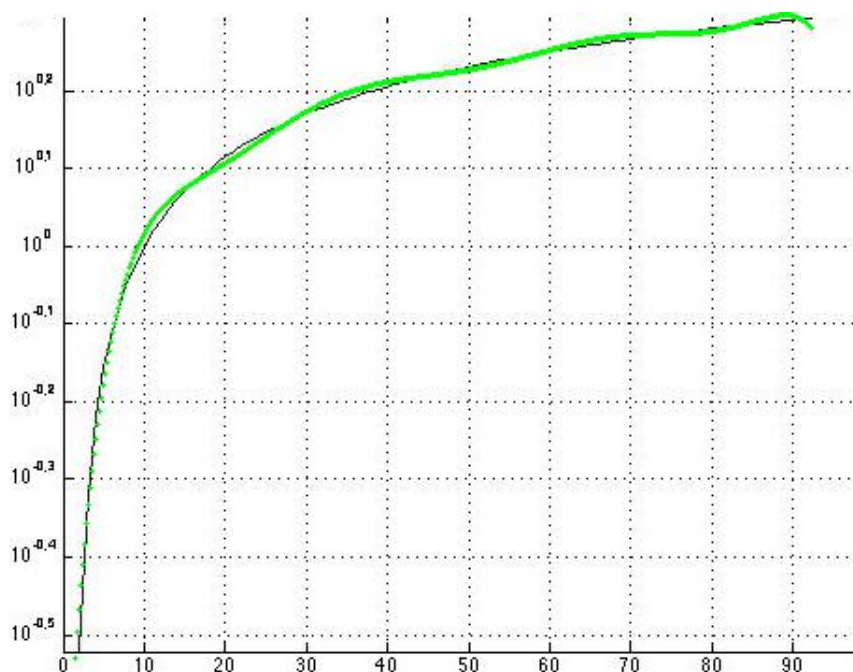


Figura 47: Ajuste com polinômio: $y = -3.07 * 10^{-16}x^8 + 5.4 * 10^{-13}x^7 - 3.97 * 10^{-10}x^6 + 1.57 * 10^{-7}x^5 - 0.00003x^4 + 0.005x^3 - 0.409x^2 + 18.1x - 98.13$

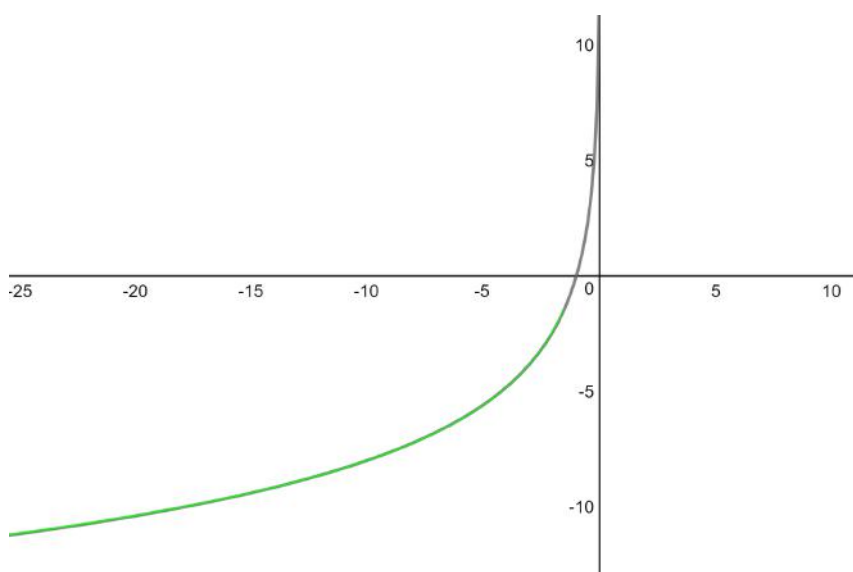


Figura 48: Ajuste com log: $y = -158 \log(-4.4x) + 832.6$

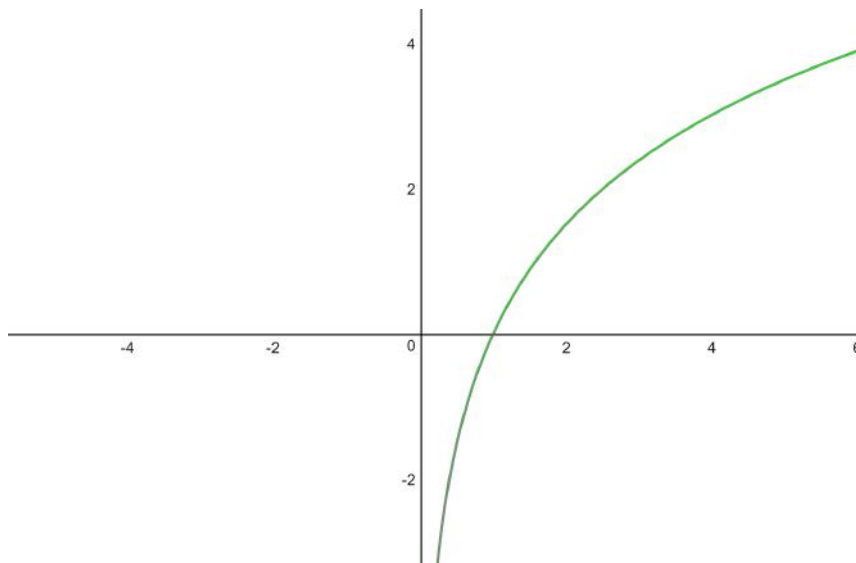


Figura 49: Ajuste com log: $y = 356.19 \log(2.69x) - 2167.28$

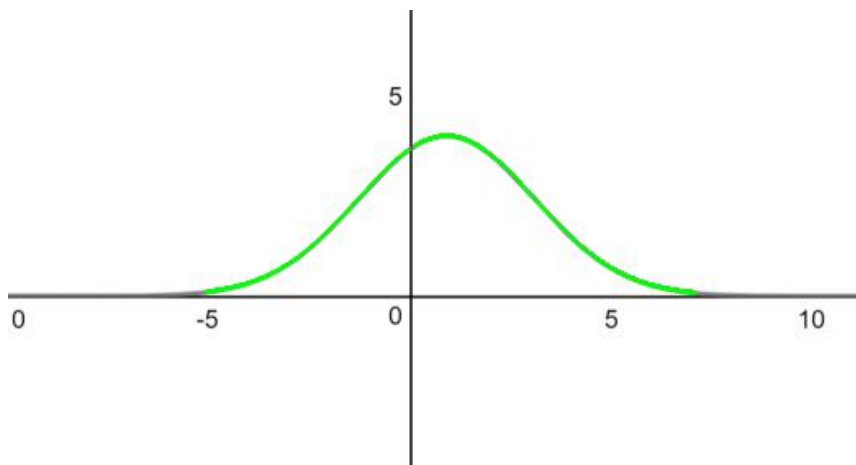


Figura 50: Ajuste com gaussiana: $y = \frac{1}{0.0024\sqrt{2\pi}} e^{-\frac{(x-155.23)^2}{59.91^2}}$

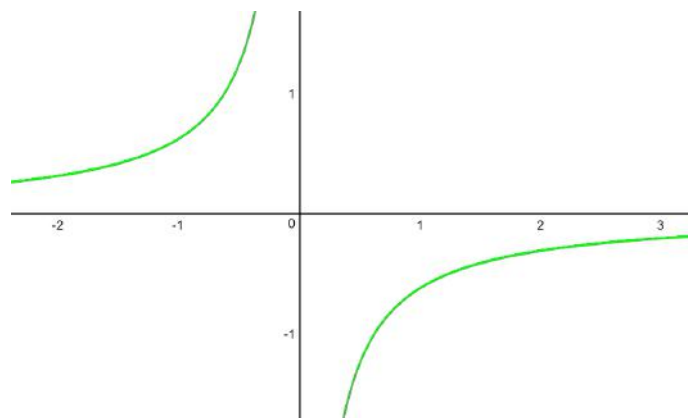


Figura 51: Ajuste com racional: $y = \frac{1}{2.47 \cdot 10^{-11}x^2 - 6.18 \cdot 10^{-5}x + 1.33 \cdot 10^{-5}}$

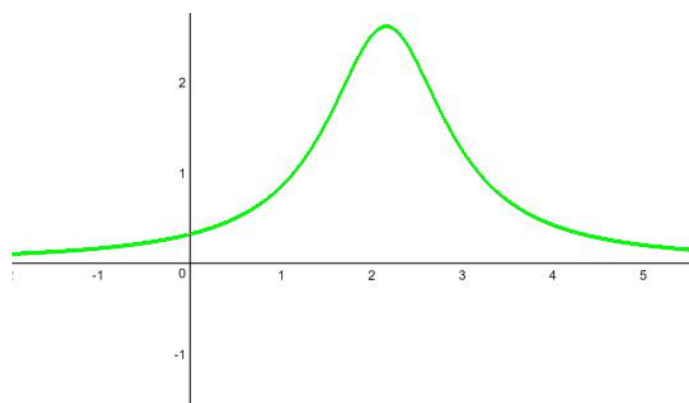


Figura 52: Ajuste com racional: $y = \frac{1}{1.65 \cdot 10^{-6}x^2 - 5.14 \cdot 10^{-4}x + 0.045}$

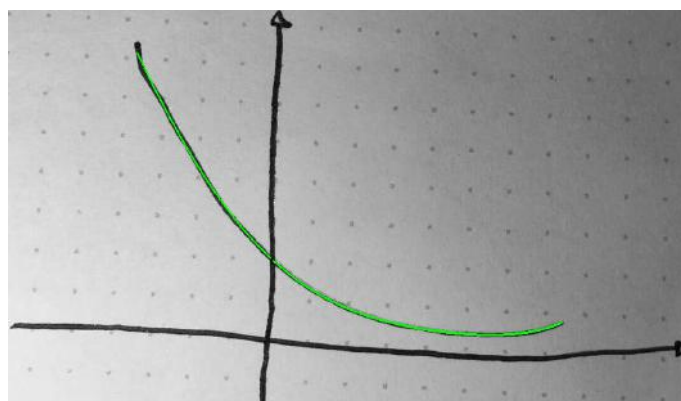


Figura 53: Ajuste com polinômio: $y = 2.3 \cdot 10^{-11}x^5 - 9.7 \cdot 10^{-9}x^4 - 3.4 \cdot 10^{-6}x^3 + 0.0033x^2 - 0.98x + 102.29$

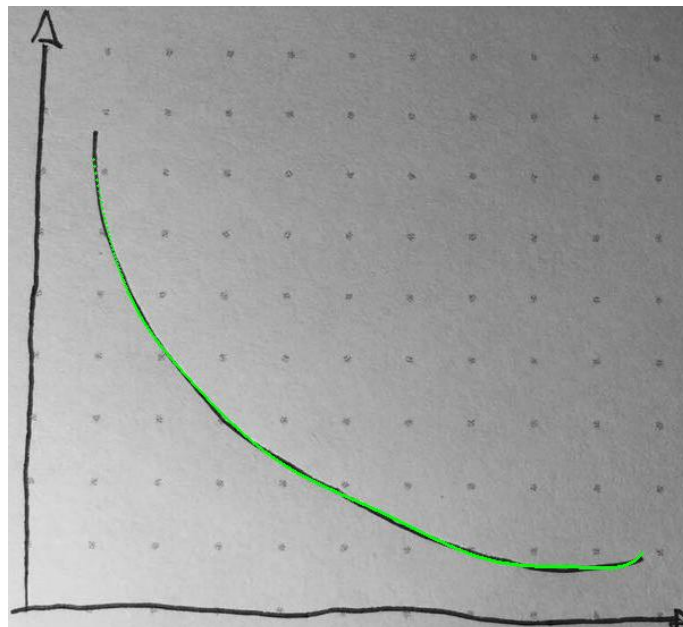


Figura 54: Ajuste com polinômio: $y = -3.07 * 10^{-16}x^8 + 5.4 * 10^{-13}x^7 - 3.97 * 10^{-10}x^6 + 1.57 * 10^{-7}x^5 - 0.00003x^4 + 0.005x^3 - 0.409x^2 + 18.1x - 98.13$

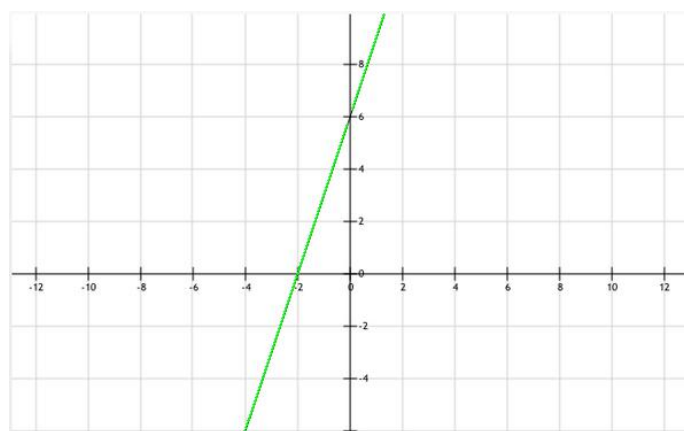


Figura 55: Ajuste com polinômio: $y = 2.99x + 134.67$

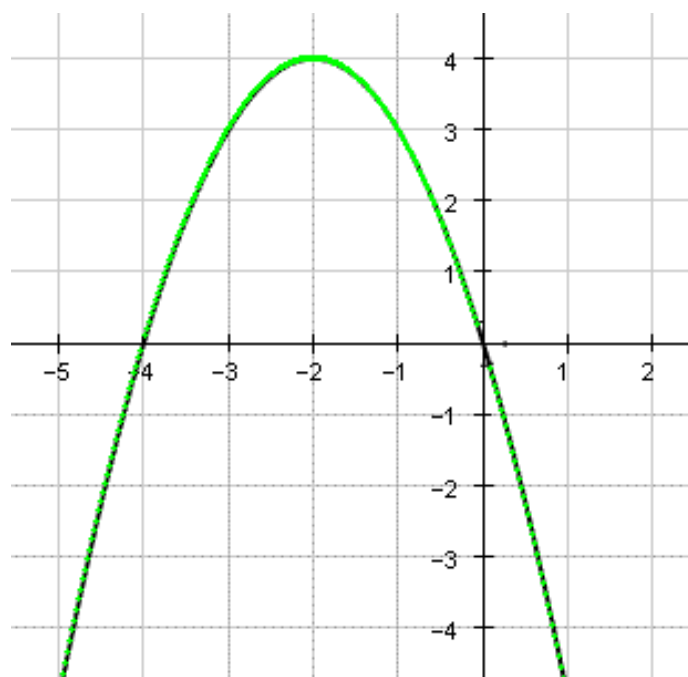


Figura 56: Ajuste com polinômio: $y = -0.021x^2 - 3.37x - 1.65$

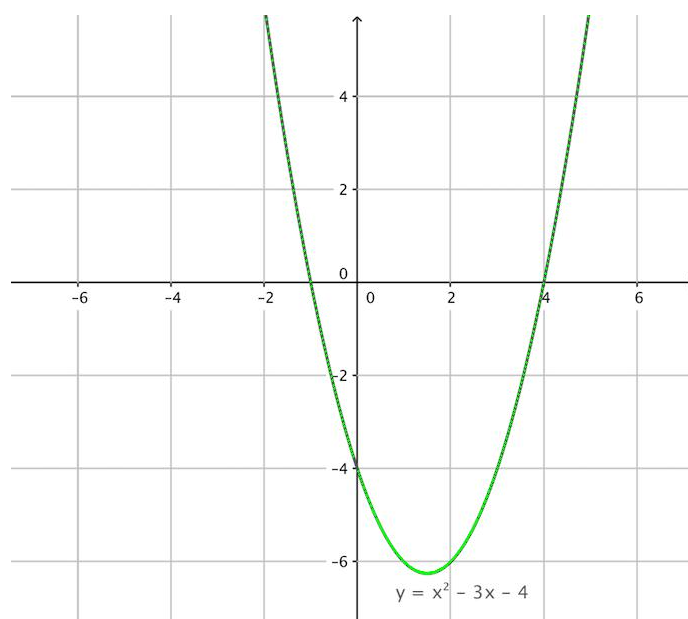


Figura 57: Ajuste com polinômio: $y = 0.021x^2 - 3.02x - 185$

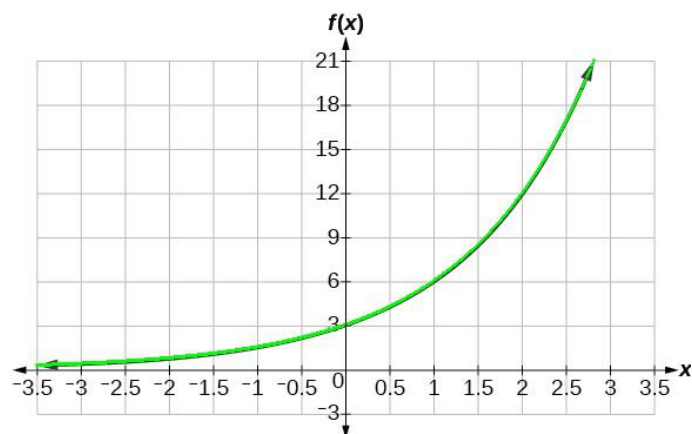


Figura 58: Ajuste com exponencial: $y = e^{0.0091*x+3.63} + 0.2$

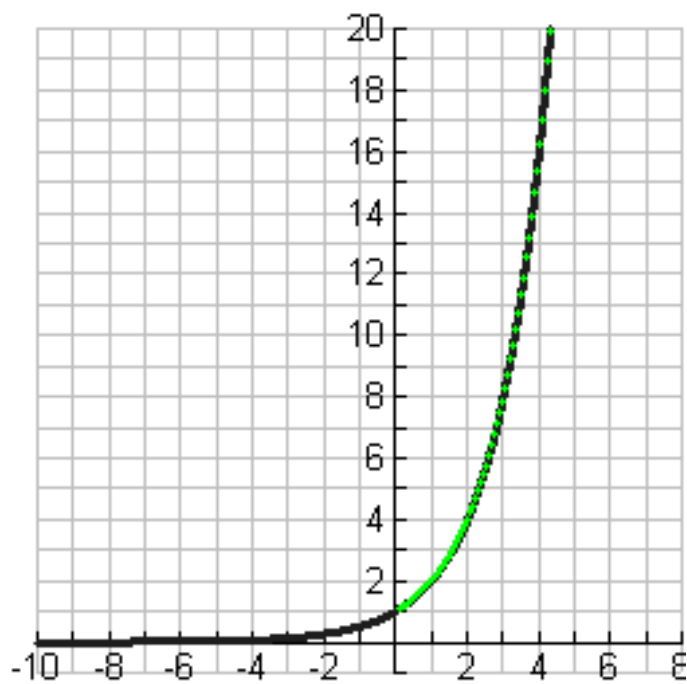


Figura 59: Ajuste com exponencial: $y = e^{0.051*x+2.44} - 1.19$

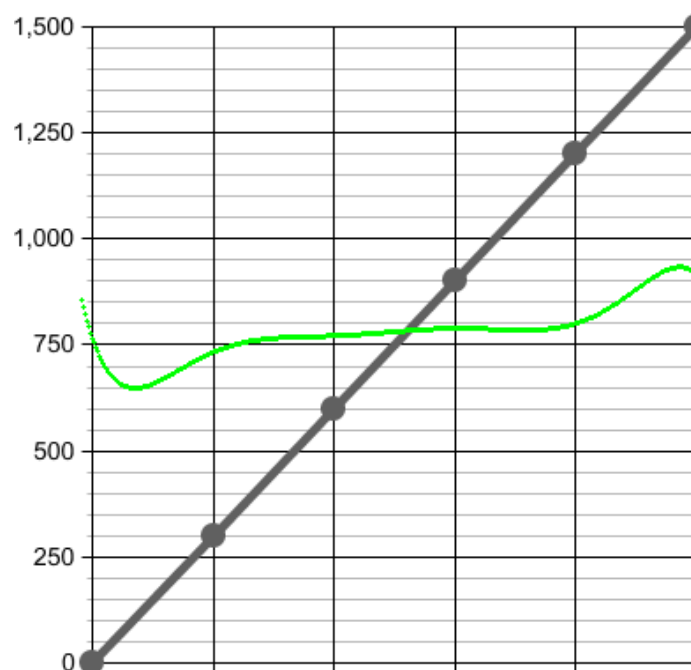


Figura 60: Ajuste com polinômio: $y = -6.7 * 10^{-14}x^7 + 1.82 * 10^{-11}x^6 + 6.37 * 10^{-10}x^5 - 3.77 * 10^{-7}x^6 + 0.000011x^3 + 0.0008x^2 + 0.039x + 65.16$

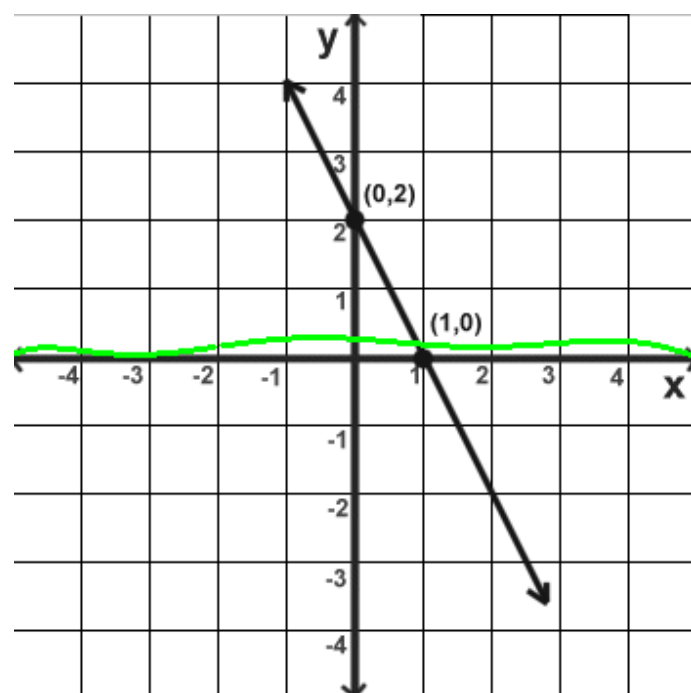


Figura 61: Ajuste com polinômio: $y = 1.47 * 10^{-14}x^7 - 9.27 * 10^{-12}x^6 + 1.43 * 10^{-9}x^5 + 9.35 * 10^{-8}x^6 - 0.000028x^3 - 0.00004x^2 + 0.14x + 39.76$

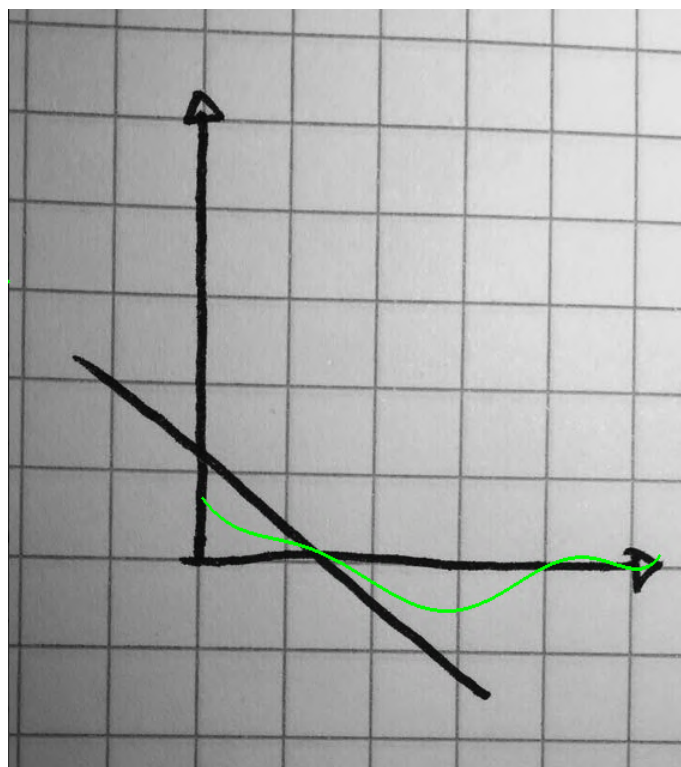


Figura 62: Ajuste com polinômio: $y = 1.65 * 10^{-14}x^7 - 1.66 * 10^{-11}x^6 + 5.26 * 10^{-9}x^5 - 2.79 * 10^{-7}x^4 - 0.00012x^3 + 0.02x^2 - 1.33x - 391.66$

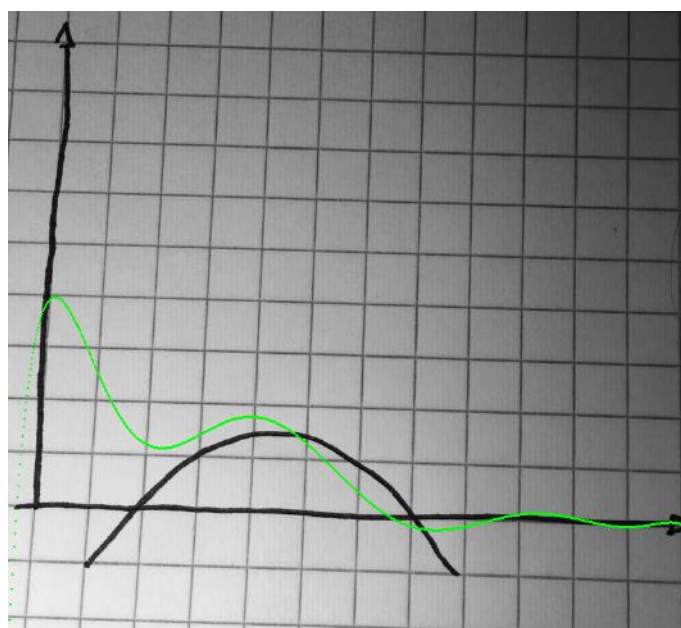


Figura 63: Ajuste com polinômio: $y = -5.26 * 10^{-18}x^8 + 4.53 * 10^{-15}x^7 - 8.84 * 10^{-14}x^6 - 7.45 * 10^{-10}x^5 + 1.09 * 10^{-7}x^4 + 0.00003x^3 - 0.005x^2 - 0.77x + 220.69$